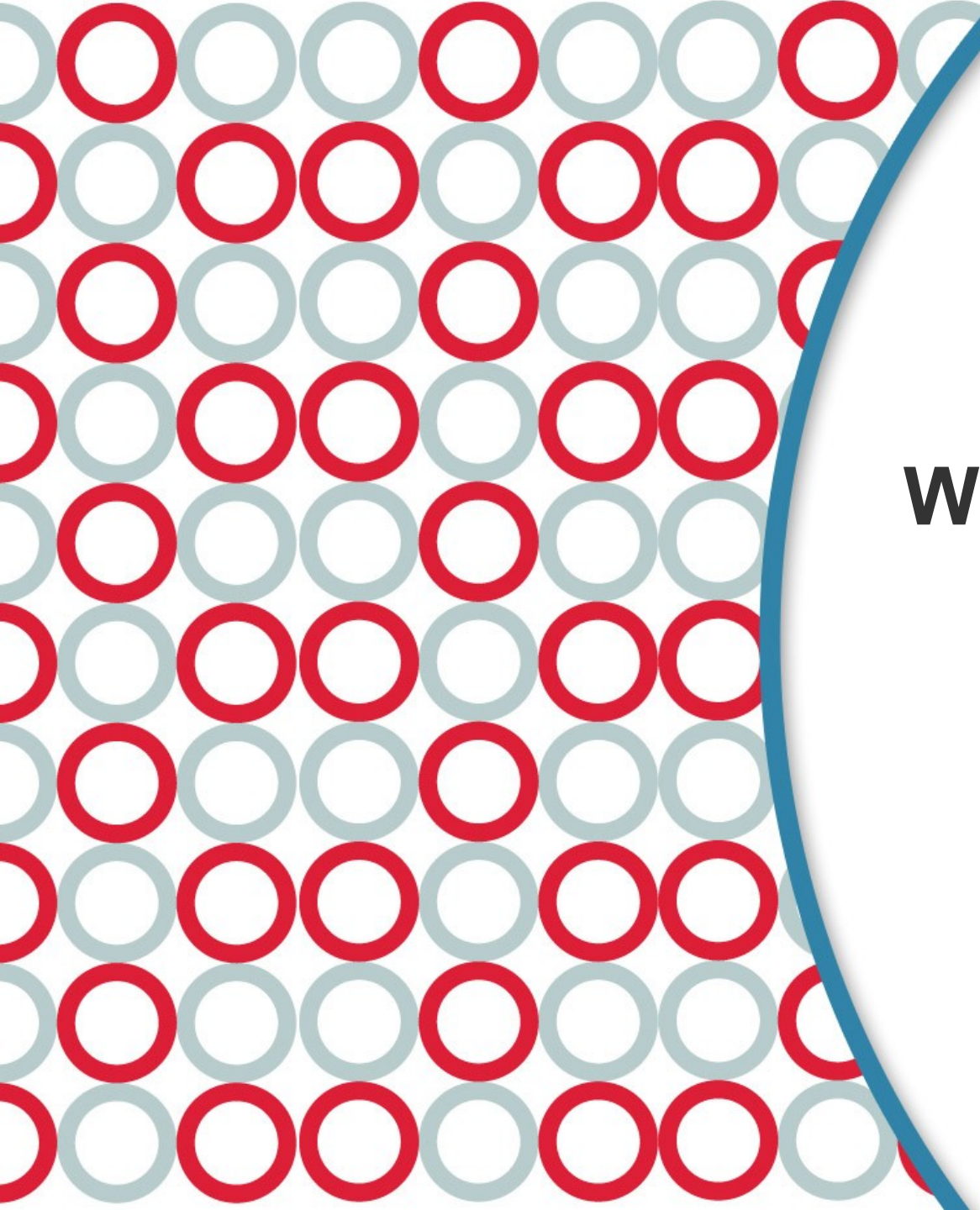


May 2019, IPT Course
Java Web Debelopment

World Wide Web

Trayan Iliev
tiliev@iproduct.org
http://iproduct.org

About me

Trayan Iliev

– CEO of IPT – Intellectual Products & Technologies

– Oracle® certified programmer 15+ Y

– end-to-end reactive fullstack apps with Java, ES6/7, TypeScript, Angular, React and Vue.js

– 12+ years IT trainer

– Voxxed Days, jPrime, jProfessionals, BGOUG, BGJUG, DEV.BG speaker

– Organizer RoboLearn hackathons and IoT enthusiast (http://robolearn.org)

Where to Find the Code?

Java Web Development projects and examples are available @ GitHub:

https://github.com/iproduct/course-java-web-development

Agenda for This Session

- WWW introduction – IP addresses, Ports, DNS, Proxy, Hosts file
- Cookies
- HTTP
- Ajax
- Practical examples

Question 1

How many of you know what is Hypermedia?

Question 3

How many know what HATEOAS stays for?

Question 2

How many people know what REST means?

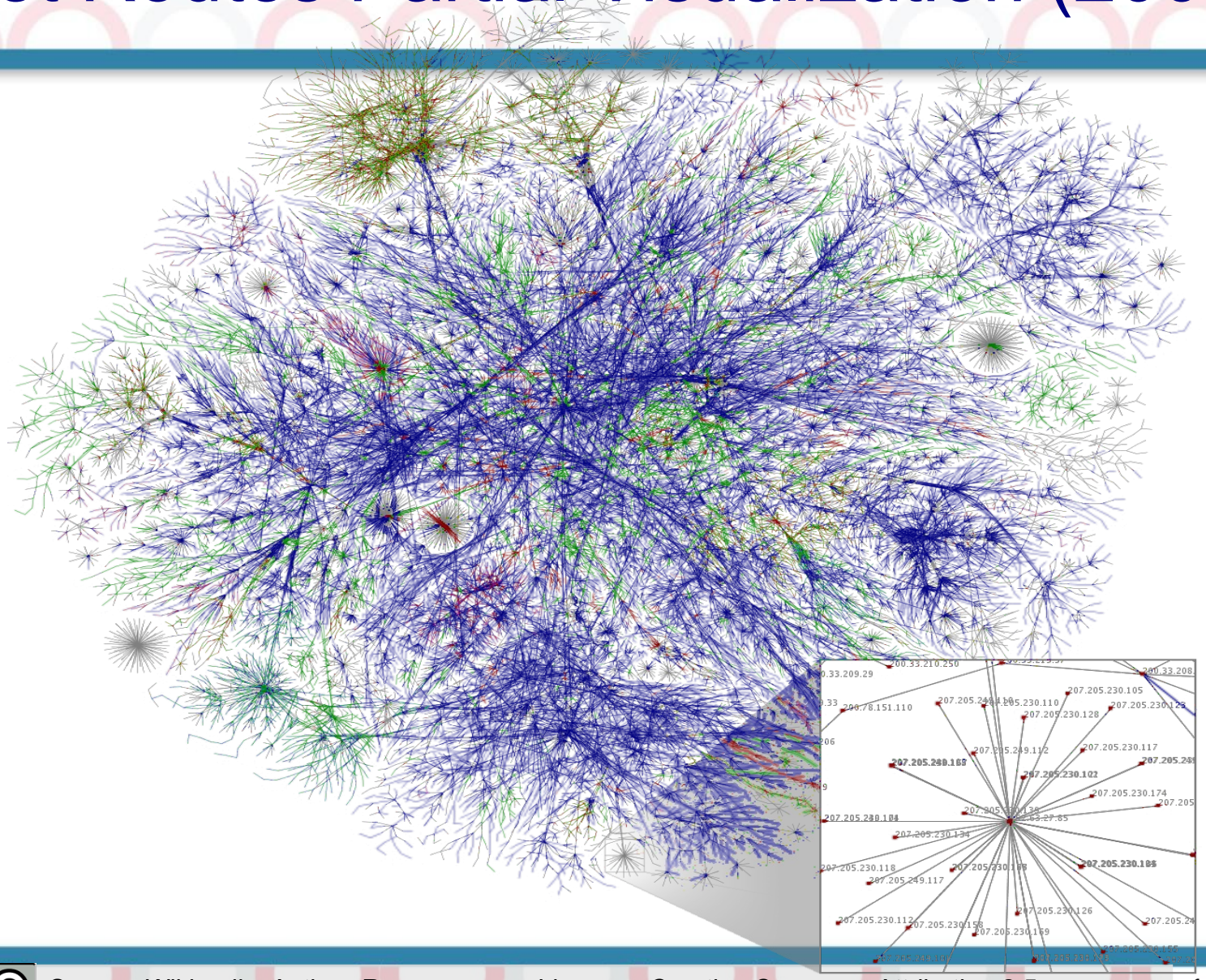
Internet (1)

- Internet – history and development:
  - ARPAnet – 1968
  - Development of a suit of protocols for global host addressing and routing – **TCP/IP** – 1973 г.
  - Divided to ARPAnet and MILNET, **Domain Name System (DNS)** – 1983 г.
  - NSFNET, high-speed T1 communication lines (1.544 Mbps) – 1986 г.
  - Internet Society – 1992 г.

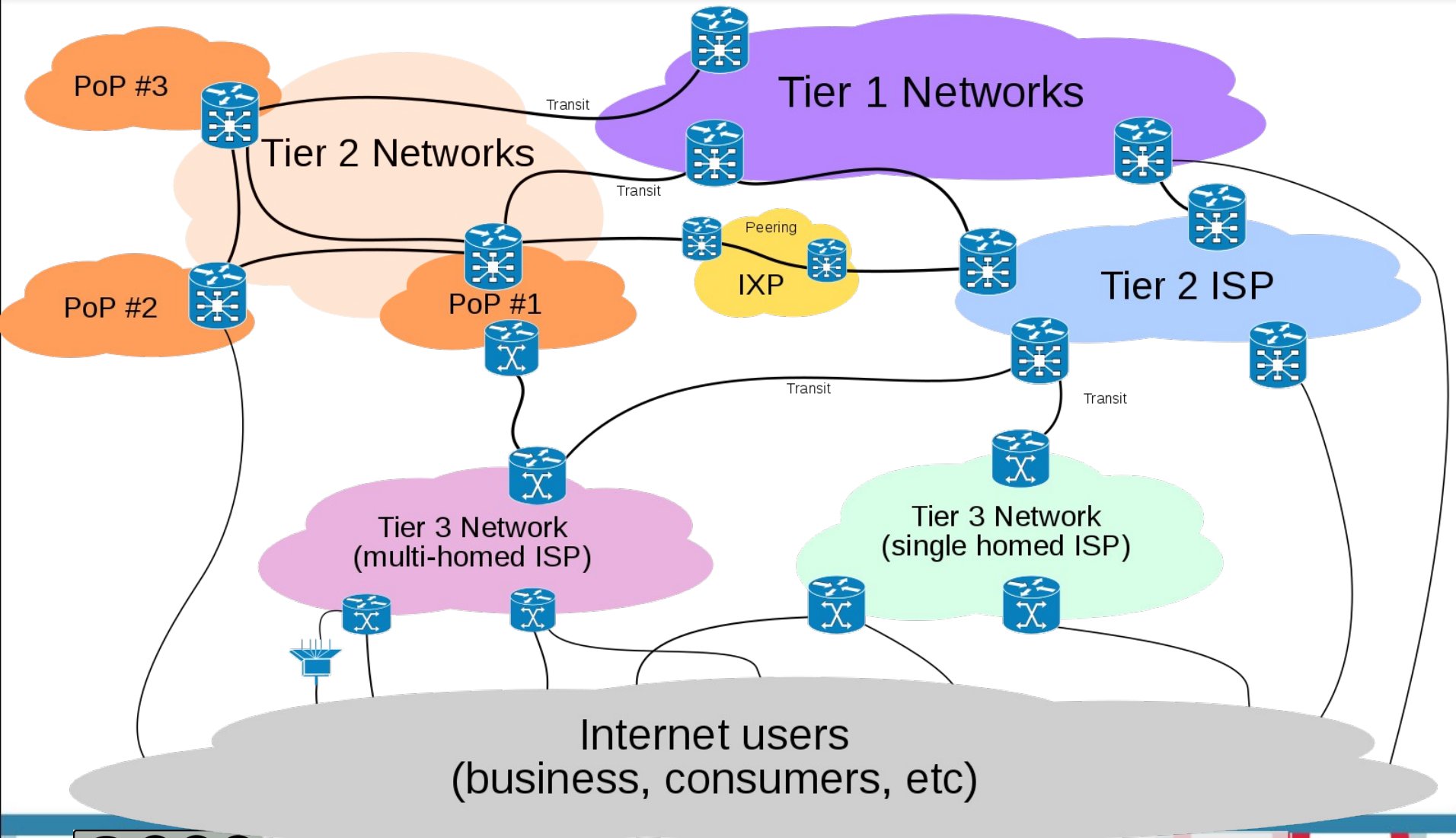
Internet (2)

- Internet Society:
  - Internet Architecture Board (IAB)
  - Internet Engineering Task Force (IETF)
  - Internet Research Task Force (IRTF)
  - Request for Comments (RFC)
- Challenge for you: find what is the purpose of RFC 1118
- Internet Corporation for Assigned Names and Numbers (ICANN)
- Perspectives – cloud computing, SaaS, PaaS, IaaS

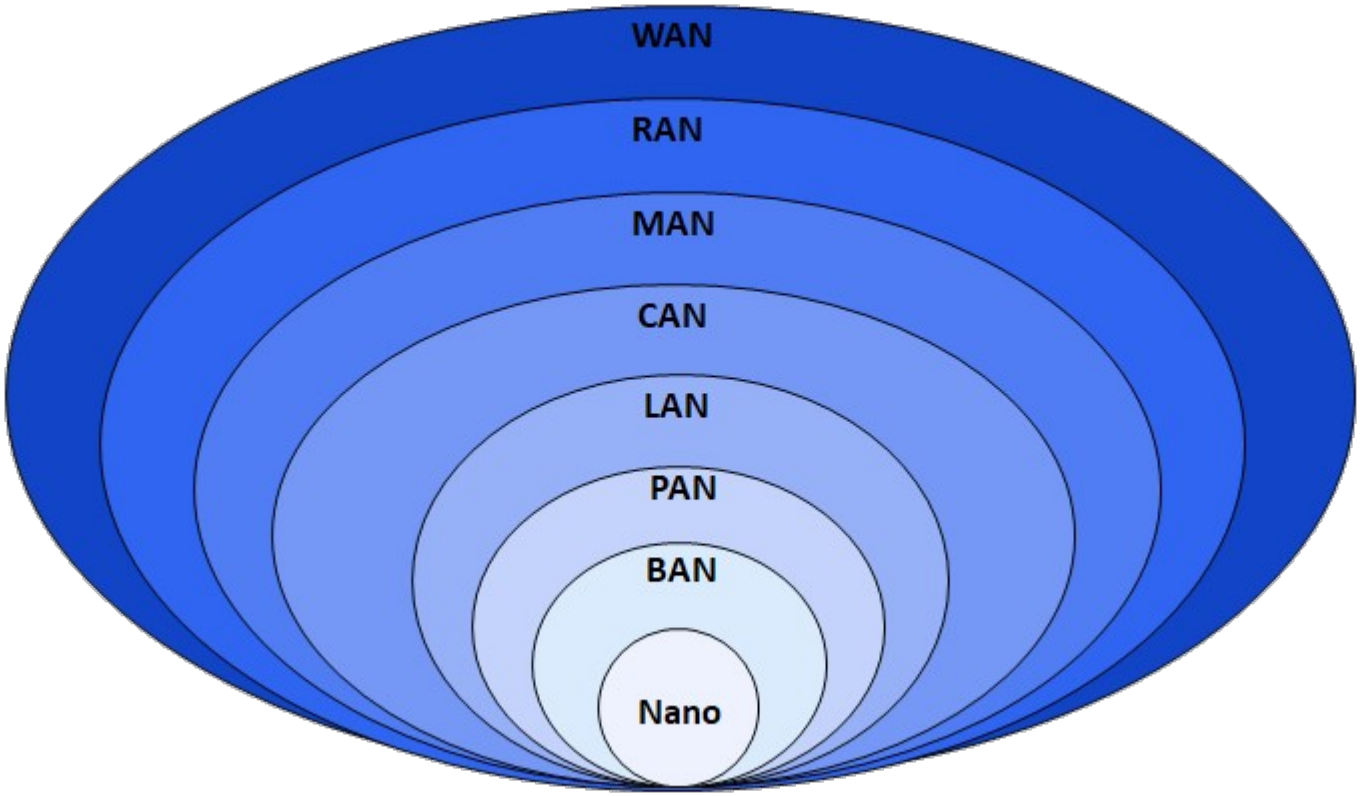
Internet Routes Partial Visualization (2005)



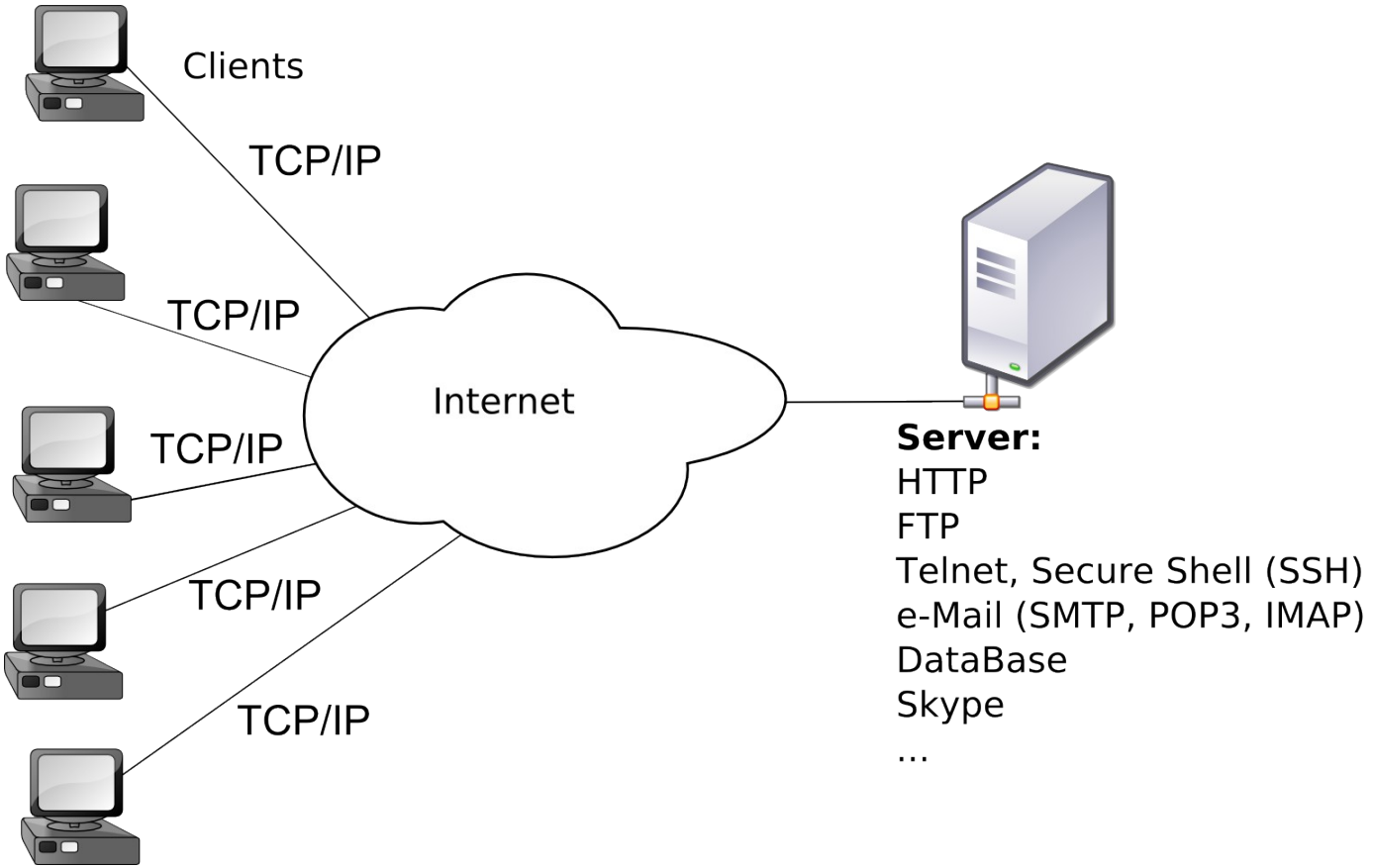
Internet Multi-Tiered Architecture



Types of Networks (by Skope)



Client-Server Architecture

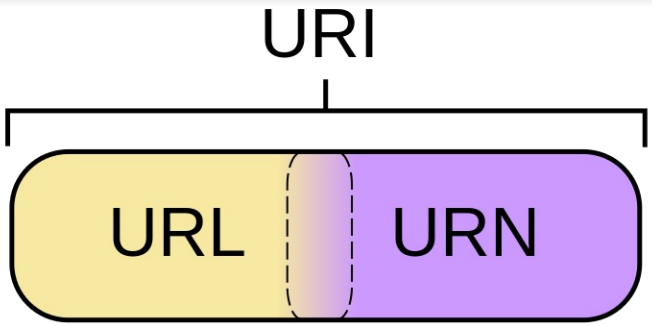


Source: Wikipedia, Author:Calimo, Лиценз: GNU Lesser General Public License v2.1, Address: http://en.wikipedia.org/wiki/File:Client-server-model.svg

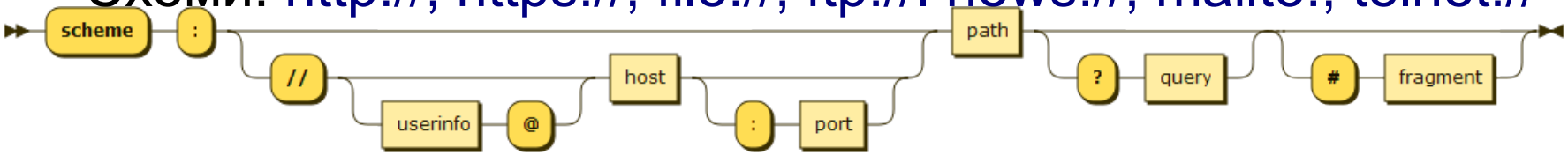
Types of Servers

- Email server
- Web server – e.g. Apache HTTPD
- Domain Name System (DNS) Server
- Database server
- File Server – e.g. FTP, SSH, or NFS
- Application Server – serving different types of web applications and services (XML, REST)

URLs, IP Addresses, and Ports



- Uniform Resource Identifier - URI:
  - Uniform Resource Locator – URL
  - Uniform Resource Name – URN
- Формат:
  scheme://domain:port/path?query_string#fragment_id
- Схеми: http://, https://, file://, ftp://. news://, mailto:, telnet://



- Пример:
  http://en.wikipedia.org/wiki/Uniform_resource_locator#Histor

URL Structure (1)

Each URL can include following elements (ordered):

- schema (protocol) type – defines how the resource will be accessed – e.g.: http, https, file, ftp, news, mailto, telnet, etc. (HOW?)
- ://
- host name (Network computer) – can be domain or IP address, defines the server computer (host) where the resource is deployed (WHERE?)
- :port number (незадължителен) – defines the service on that host (e.g. Web server is on port 80 by default)
- full path to the resource on the server – for example: **/wiki/Uniform_resource_locator** (WHAT?)

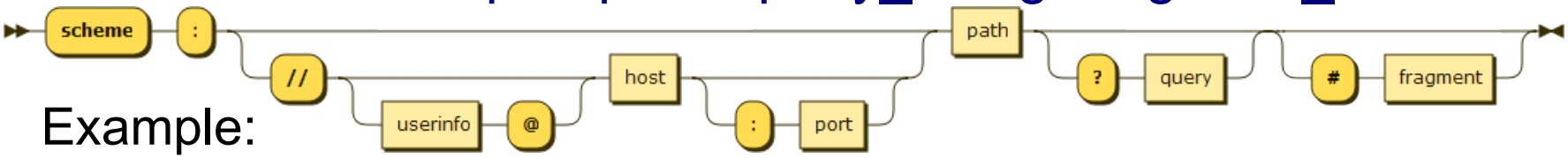
URL Structure (2)

URLs pointing to **dynamic resources** (CGI scripts, Java Servlet / JSP, ASP, PHP, etc.) often include:

- **?**list of request query parameters (query string) – e.g.: ?courseId=1211&role=student
- **#**fragment identifier – defines the fragment (part) of the resource we want to access, used by asynchronous javascript page loading (AJAX) applications to encode the local page state – e.g. #view=fitb&nameddest=Chapter3

The wole syntax is:

scheme://domain:port/path?query_string#fragment_id



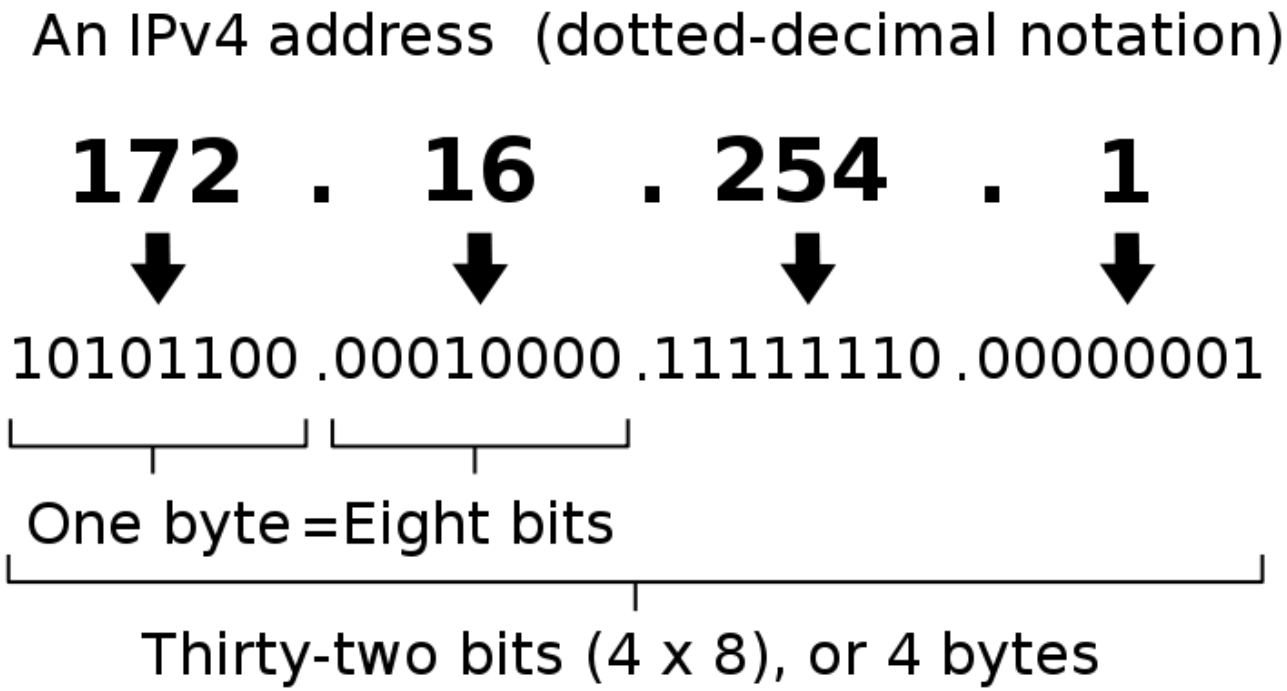
Example:

http://en.wikipedia.org/wiki/Uniform_resource_locator#History

Relative URI/URLs (URI References)

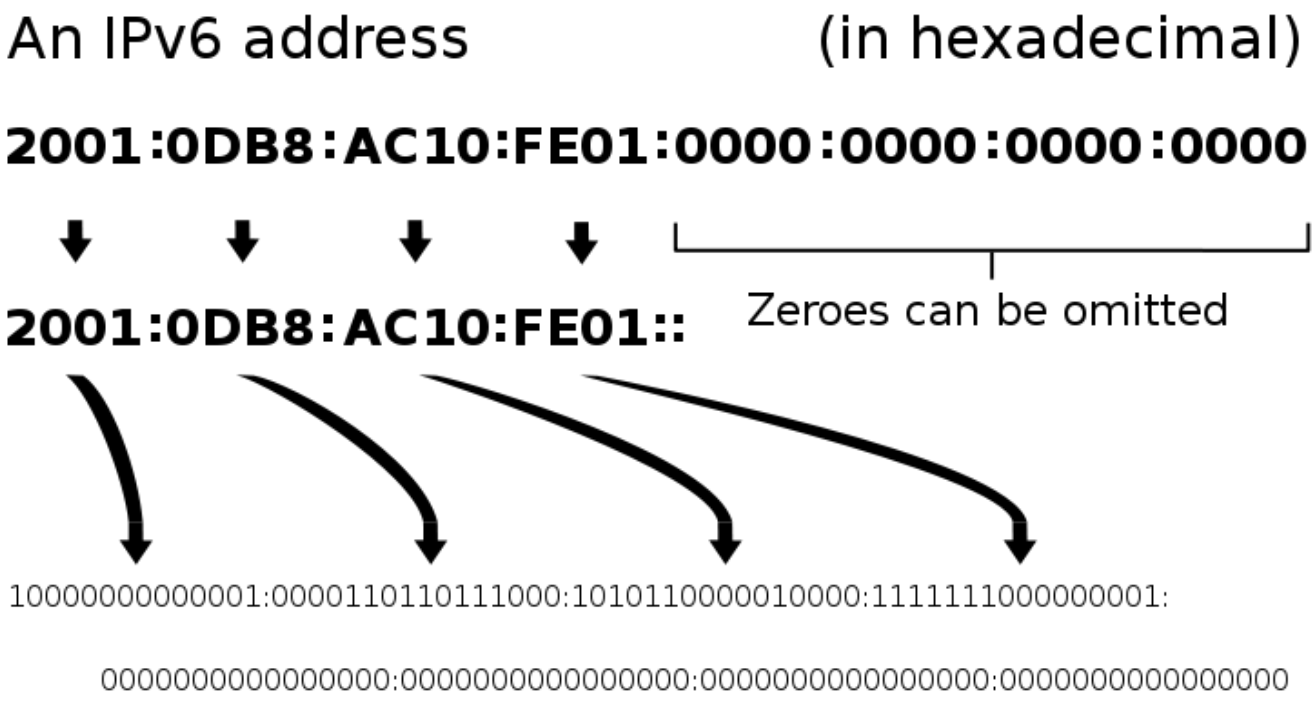
- URLs can be **absolute** (by fully specifying how to access the resource) and **relative** (defining only the differences from the currently accessed resource URL)
- Examples for relative URLs: (http://en.wikipedia.org/wiki/Uniform_resource_identifier):
- //example.org/scheme-relative/URI/with/absolute/path/to/resource.txt
- /relative/URI/with/absolute/path/to/resource.txt
- relative/path/to/resource.txt
- ../../../resource.txt
- ./resource.txt#frag01
- resource.txt
- #frag01

IP Version 4 Addresses



Источник: Wikipedia, Автор: Indeterminate, Лиценз: public domain, адрес: http://en.wikipedia.org/wiki/File:Ipv4_address.svg

IP Version 6 Addresses



Източник: Wikipedia, Автор: Indeterminate, Лиценз: public domain,
Адрес: http://en.wikipedia.org/wiki/File:Ipv6_address.svg

Well Known Ports – TCP and UDP

...

67/UDP Bootstrap Protocol (BOOTP) Server; also used by Dynamic Host Configuration Protocol (DHCP) Official

68/UDP Bootstrap Protocol (BOOTP) Client; also used by Dynamic Host Configuration Protocol (DHCP) Official

69/UDP Trivial File Transfer Protocol (TFTP) Official

70/TCP Gopher protocol Official

79/TCP Finger protocol Official

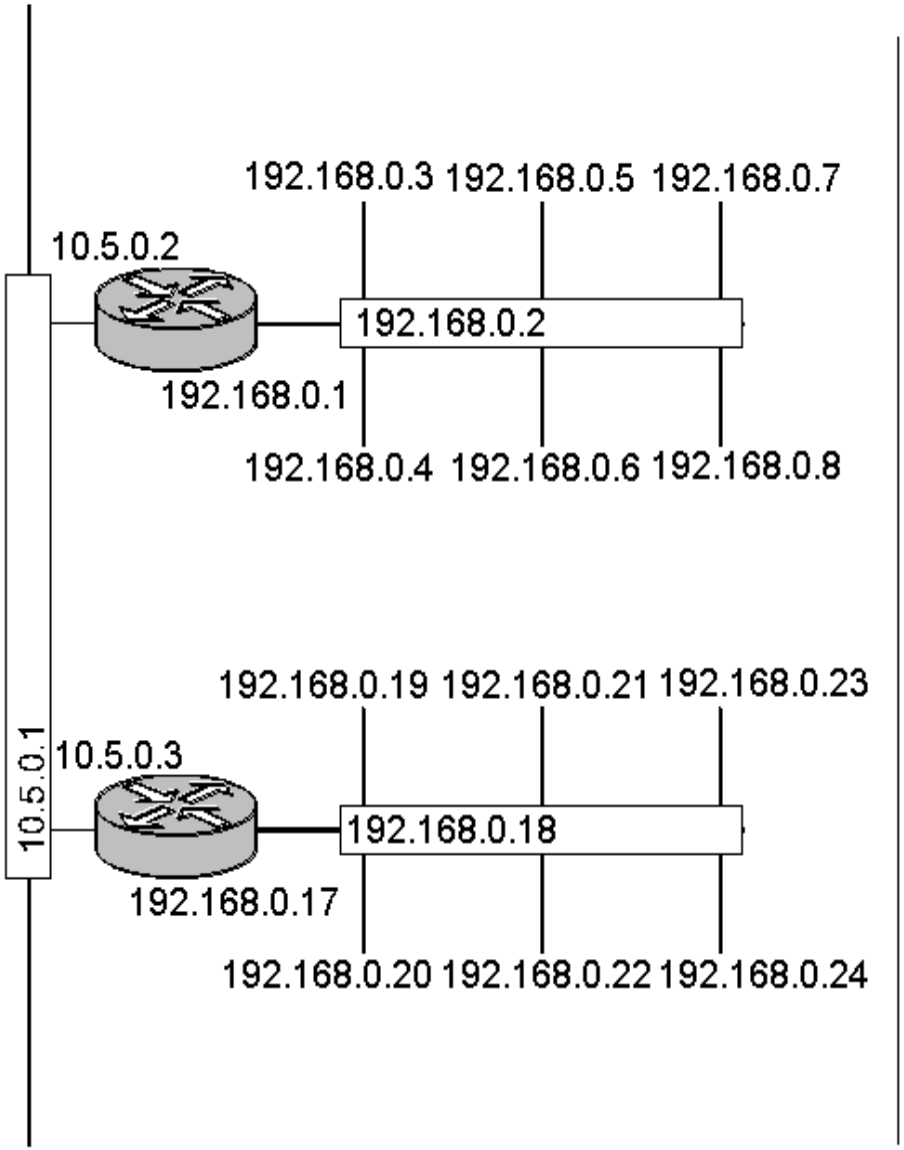
80/TCP Hypertext Transfer Protocol (HTTP) Official

81/TCP Torpark—Onion routing Unofficial

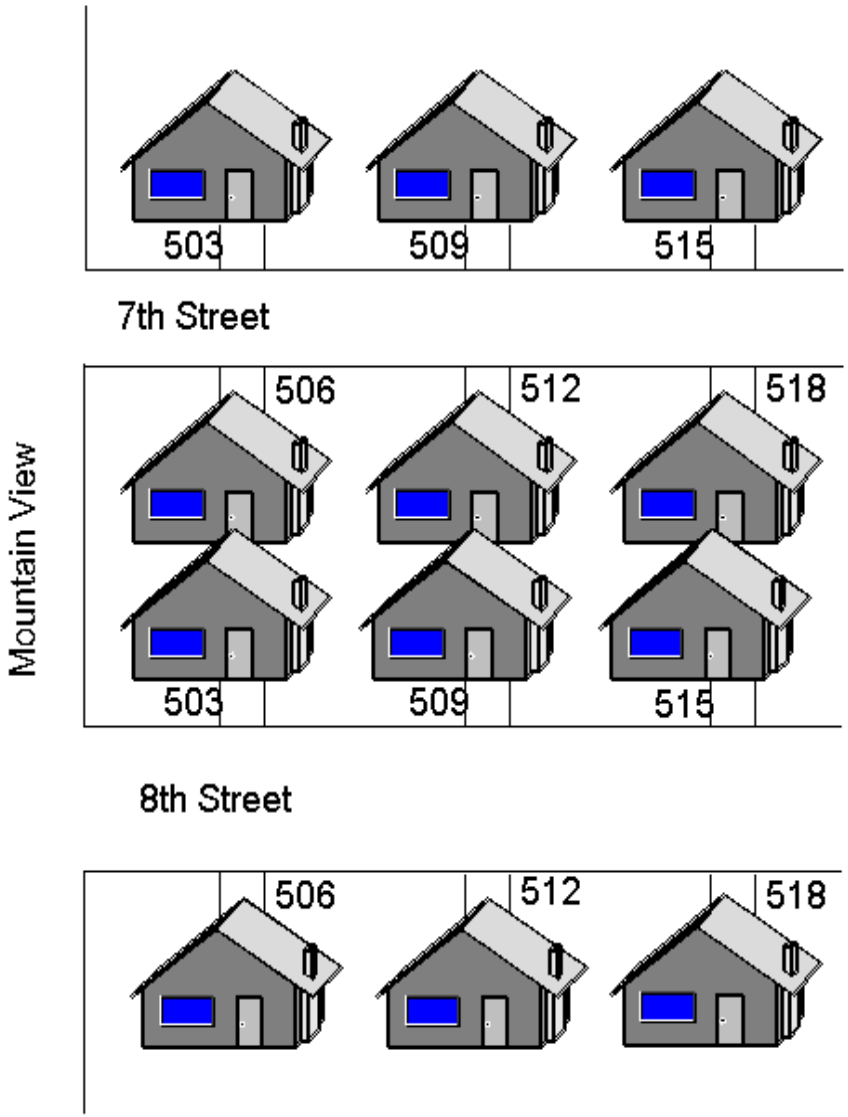
82/UDP Torpark—Control Unofficial

83/TCP MIT ML Device Official

88/TCP Kerberos—authentication system Official

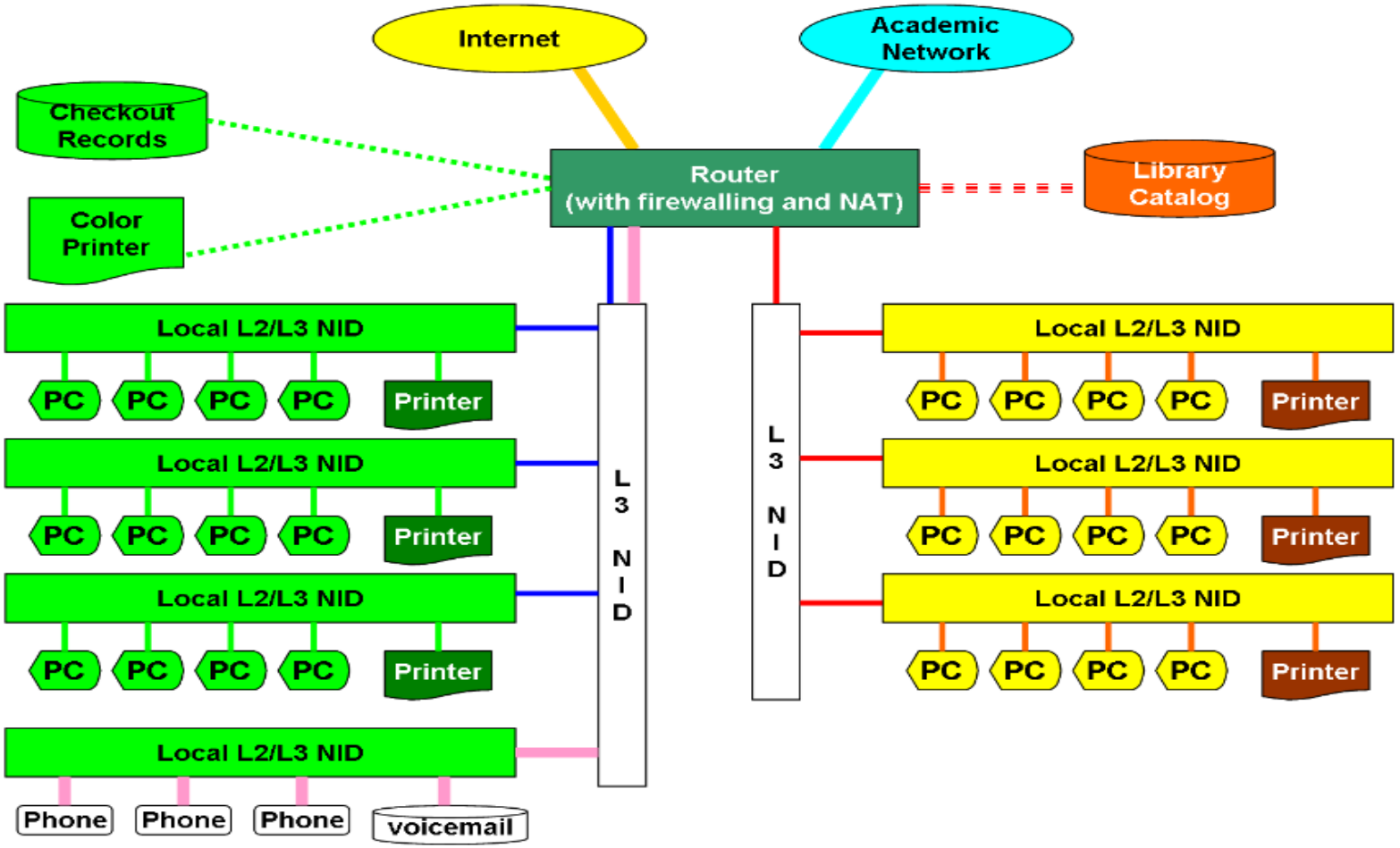


Network



Neighborhood

Network Topology (Wikipedia)



OSI Model

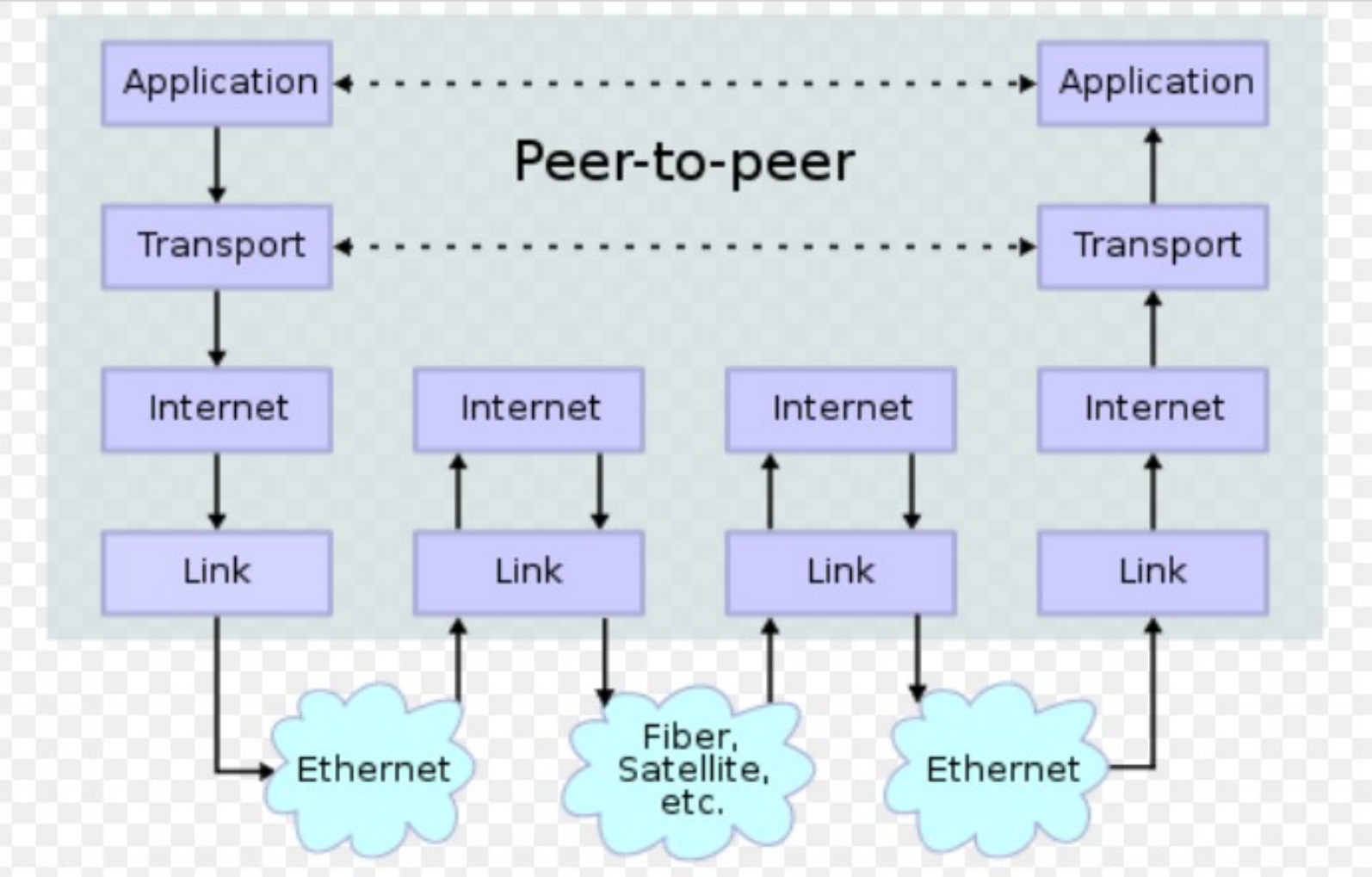
- OSI = Open Systems Interconnect Basic Reference

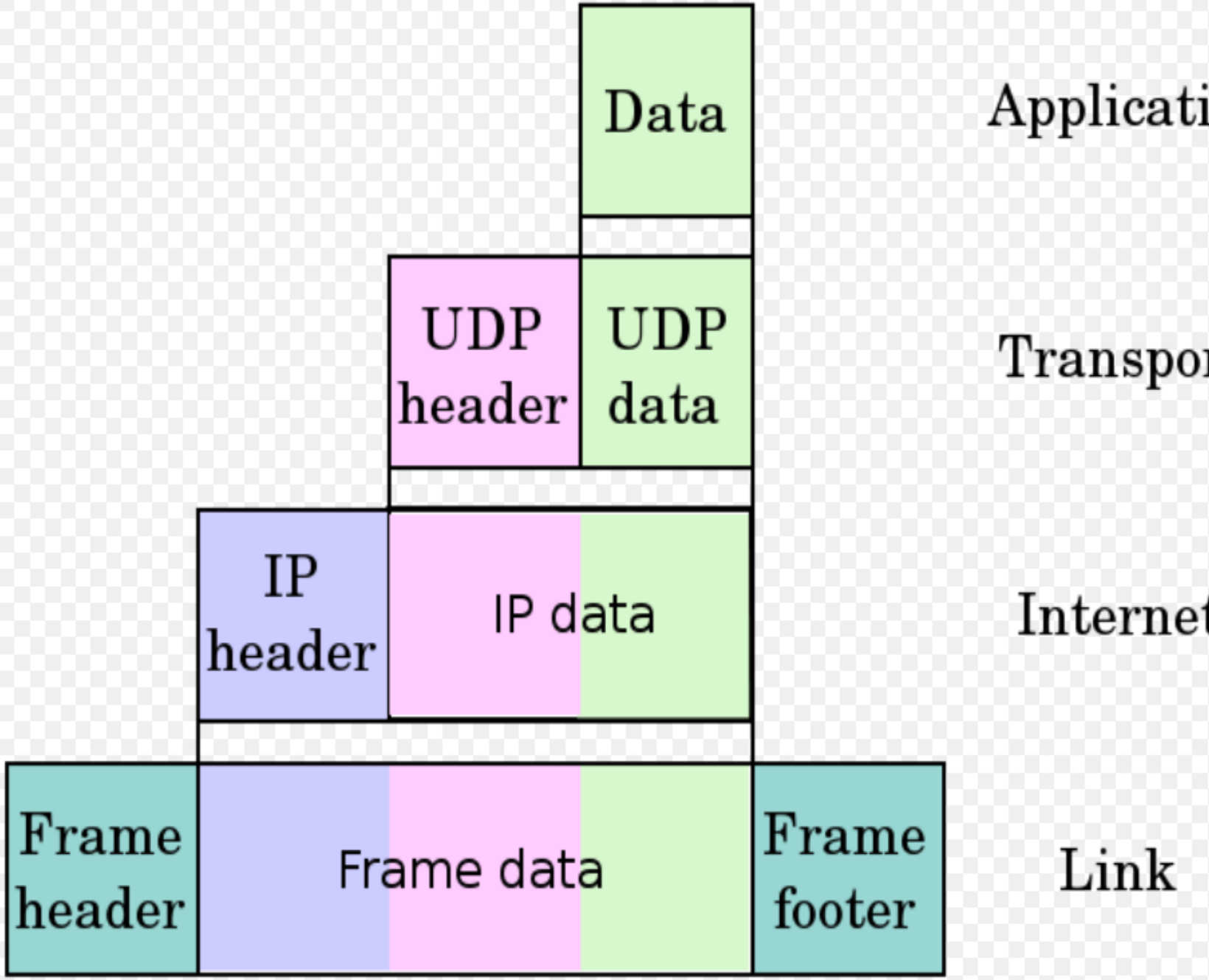
OSI модел

Application	Приложен слой	layer 7
Presentation	Представителен слой	layer 6
Session	Сесиен слой	layer 5
Transport	Транспортен слой	layer 4
Network	Мрежови слой	layer 3
DataLink	Канален слой	layer 2
Physical	Физически слой	layer 1

OSI Model			
	Data unit	**Layer**	**Function**
Host layers	Data	7. Application	Network process to application
		6. Presentation	Data representation and encryption
		5. Session	Interhost communication
	Segment	4. Transport	End-to-end connections and reliability
Media layers	Packet	3. Network	Path determination and logical addressing
	Frame	2. Data Link	Physical addressing
	Bit	1. Physical	Media, signal and binary transmission

TCP/IP Protocol Stack (Wikipedia)

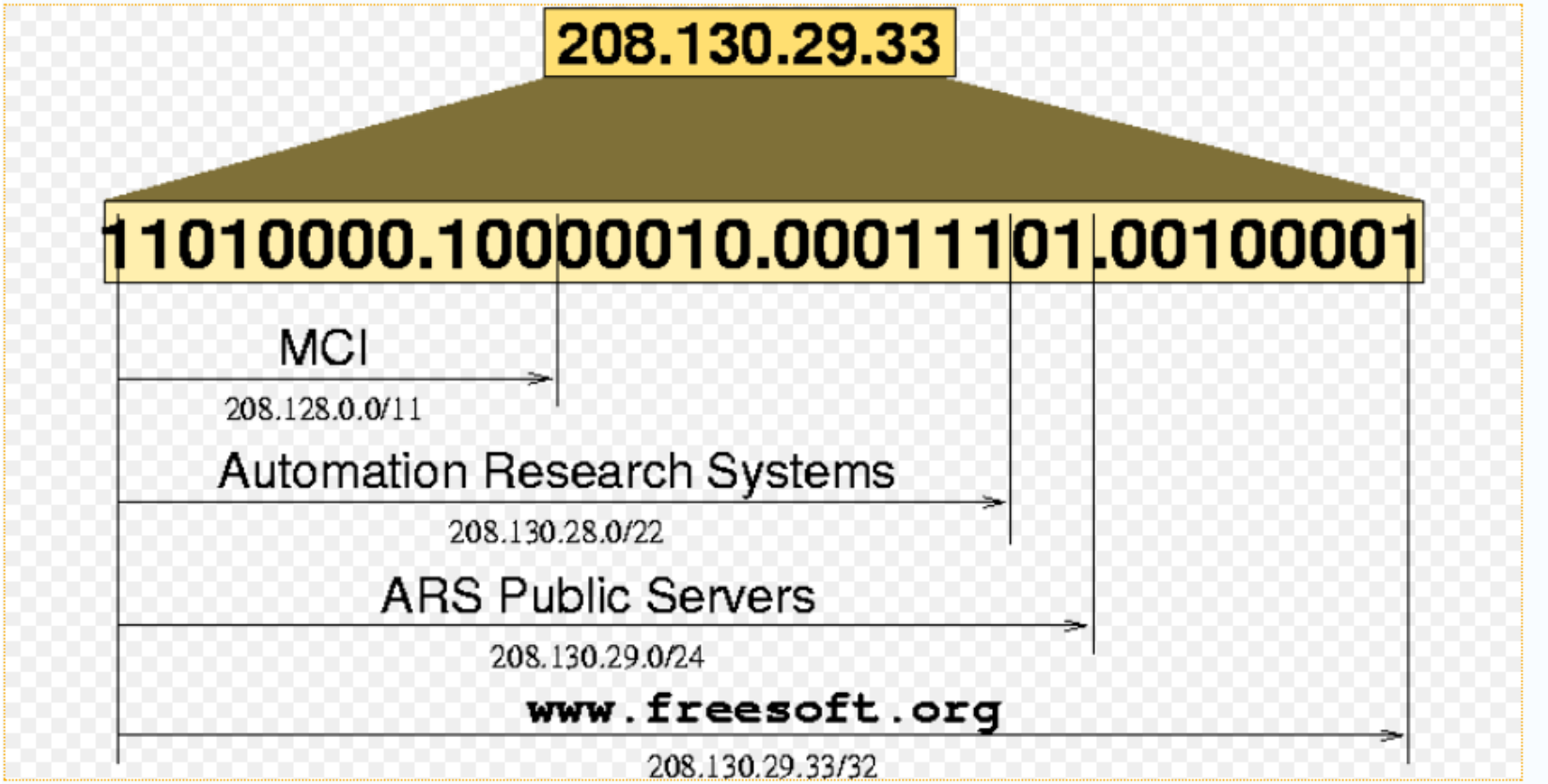



Data
Application
UDP header
UDP data
Transport
IP header
IP data
Internet
Frame header
Frame data
Frame footer
Link

IPv4 – Address Classes (Wikipedia)

Class	Leading Bits	Size of *Network Number* Bit field	Size of *Rest* Bit field	Number of Networks	Hosts per Network
Class A	0	8	24	128	16,777,214
Class B	10	16	16	16,384	65,534
Class C	110	24	8	2,097,152	254
Class D (multicast)	1110	not defined	not defined	not defined	not defined
Class E (reserved)	1111	not defined	not defined	not defined	not defined

Classless Inter-Domain Rooting (CIDR)



Transfer Control Protocol – TCP

- Reliable data transfer

+	0 - 3	4 - 9	10 - 15	16 - 31
0	Порт на източника			Порт на получателя
32	Номер по ред			
64	Сегментен номер			
96	Дължина на заглавието (хедъра)	Запазен	Кодови (за синхронизация)	Големина на рамката
128	Сума за проверка			Указател за спешност
160	Опции и пълнеж			
192	Данни			

User Datagram Protocol – UDP

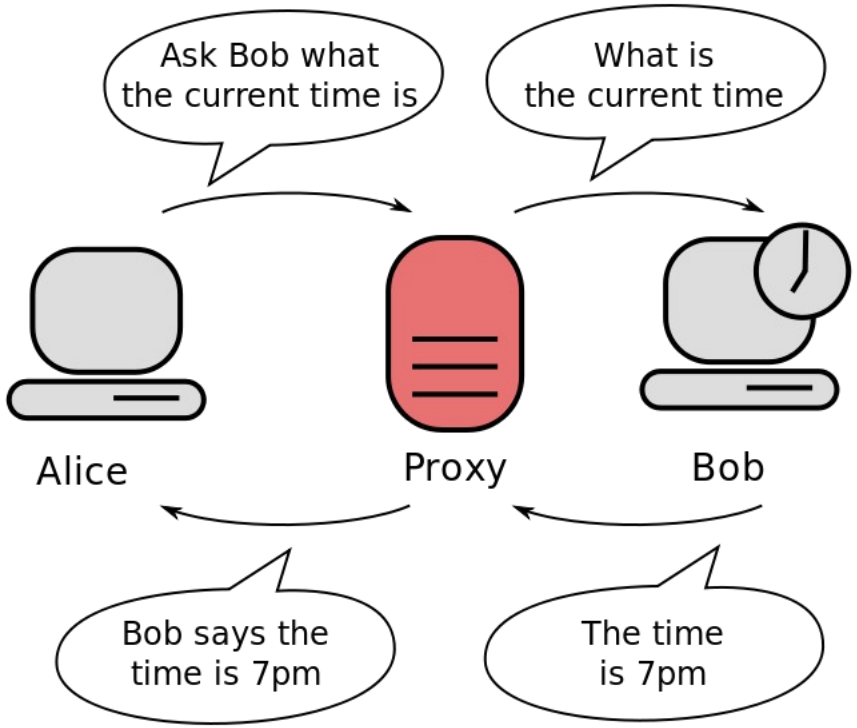
- UDP fast, but not reliable out of the box

+	Битове от 0 - 15	Битове 16 - 31
0	Изходен Порт	Порт на дестинацията
32	Дължина	Контролна сума
64	Данни	

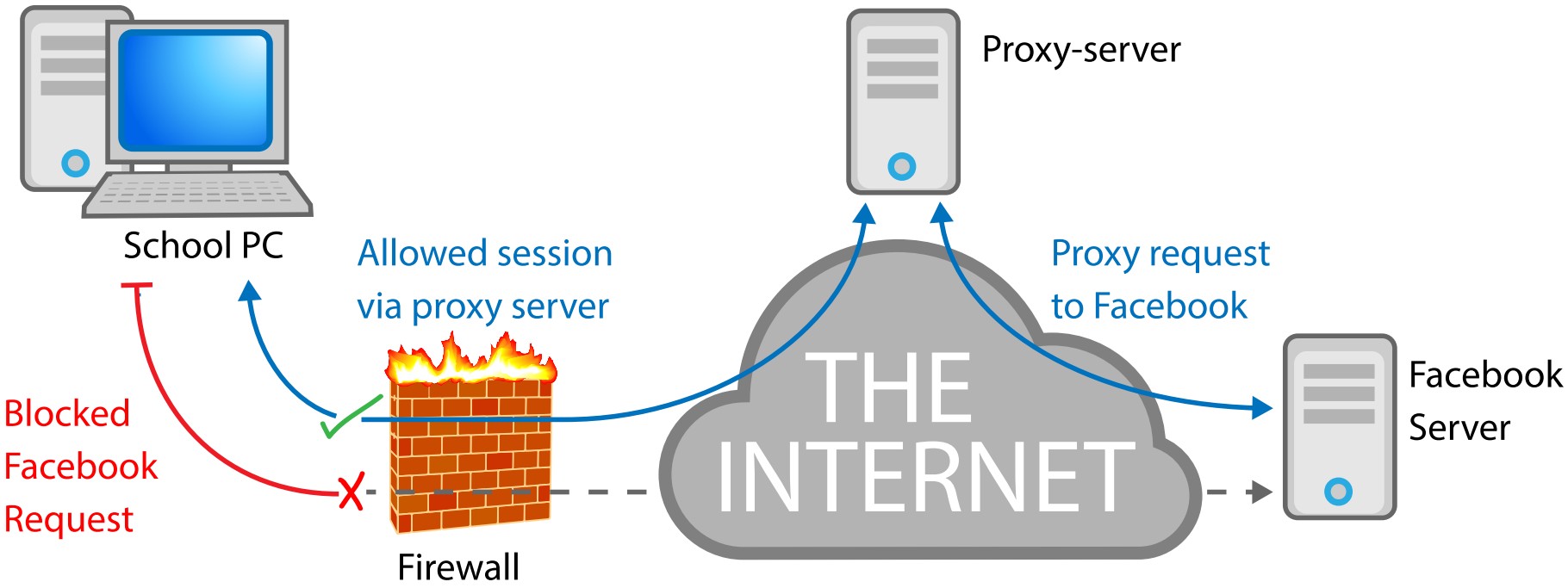
Well Known Ports – TCP и UDP

...
7/TCP,UDP Echo Official
...
20/TCP FTP - data Official
21/TCP FTP—control (command) Official
22/TCP,UDP Secure Shell (SSH)—used for secure logins, file transfers (scp, sftp) and port forwarding Official
23/TCP Telnet protocol—unencrypted text communications Official
25/TCP Simple Mail Transfer Protocol (SMTP)—used for e-mail routing between mail servers Official
...
80/TCP Hypertext Transfer Protocol (HTTP) Official
...
88/TCP Kerberos—authentication system Official
...

Web Proxy



Firewall



World Wide Web (WWW) Service

- World Wide Web (WWW) or W3 is a system of mutually connected hypertext documents (resources), accessible using the Internet
- Today World Wide Web is one of the main Internet services to the extent that the two terms are often used as synonyms

The NeXT computer used by Tim Berners-Lee in CERN. The label says: „The machine is a server. DO NOT POWER DOWN!!“

World Wide Web (WWW) – Main Concepts

- The idea for World Wide Web is suggested by Tim Berners-Lee in 1989 in CERN.
- Documents in World Wide Web, called ***web pages***, могат can contain *text, images, video, and other multimedia components*, and the connections between them are specified using *hyperlinks*.
- ***Web Sites*** include multiple connected *web pages* for a specific purpose
- They are **deployed** on a ***Web Server*** and are accessed using ***Web Client*** (*web browser – IE, Mozilla, Chrome*), using a protocol called: ***Hypertext Transfer Protocol (HTTP)***

Media Types. Multimedia

- Text – a linear sequence of character data
- Graphics – raster and vector images
- Animation – sequence of changing images (frames) with different frame-rates
- Audio – can be discretized analog signal (e.g. MP3) or just musical notes (MIDI)
- Video – different file formats, can be linear or interactive
- 3D Graphics / Animation – using 3D modelling and rendering techniques to achieve realistic, real-world like visualization
- And more – haptic feedback, smells, ...
- Multimedia – combining different media types in a coherent and consitent way to achieve better/more realistic user experience

HTTP Request Structure

GET */context/Servlet* **HTTP/1.1**

Host: *Client_Host_Name*

Header2: *Header2_Data*

...

HeaderN: *HeaderN_Data*

<Празен ред>

POST */context/Servlet* **HTTP/1.1**

Host: *Client_Host_Name*

Header2: *Header2_Data*

...

HeaderN: *HeaderN_Data*

<Празен ред>

POST_Data

HTTP Response Structure

HTTP/1.1 200 OK

Content-Type: text/html

Header2: *Header2_Data*

...

HeaderN: *HeaderN_Data*

<Празен ред>

<!DOCTYPE
Document_Type
*_Definition***>**

```
<html>
    <head>
        <title>...</title>
    </head>
    <body>
        ...
    </body>
</html>
```

HTTP Response Status Codes

- 100 Continue
- 101 Switching Protocols
- 200 OK
- 201 Created
- 202 Accepted
- 203 Non-Authoritative Information
- 204 No Content
- 205 Reset Content
- 301 Moved Permanently
- 302 Found
- 303 See Other
- 304 Not Modified
- 307 Temporary Redirect
- 400 Bad Request
- 401 Unauthorized
- 403 Forbidden
- 404 Not Found

HTTP Response Status Codes

- **405 Method Not Allowed**
- **415 Unsupported Media Type**
- **417 Expectation Failed**
- **500 Internal Server Error**
- **501 Not Implemented**
- **503 Service Unavailable**
- **505 HTTP Version Not Supported**

Types of Web Sites

- Static Web sites – show the same information for all visitors – can include hypertext, images, videos, navigation menus, etc.
- Dynamic Web sites – change and tune the content according to the specific visitor
  - server-site – use server technologies for dynamic web content (page) generation (data comes drom DB)
  - client-side – use JavaScript and asynchronous data actualization

Basic Recommendations (1)

- Define who are your users and what are their main goals and objectives using the web site. You can use the *persona modeling technique* to for this purpose.
- Align user and business goals
- Use each media according to its purpose – don't just copy you printed brochure
- Emphasize the functionality instead of fun
- Start simple and focuse on whats important Then gradually extend.

Basic Recommendations (2)

- Concentrate on users and their goals
- Implement intuitive navigation paths – design navigation before implementing the site
- Follow the web conventions and the “principle of least astonishment (POLA)”:
  - Navigation system
  - Interface metaphores
  - Visual layout of elements
  - Color conventions
- More concrete recommendations you can find at Jakub Linowski»s site: **http://goodui.org**

Comparison between HTML and XML

HTML:

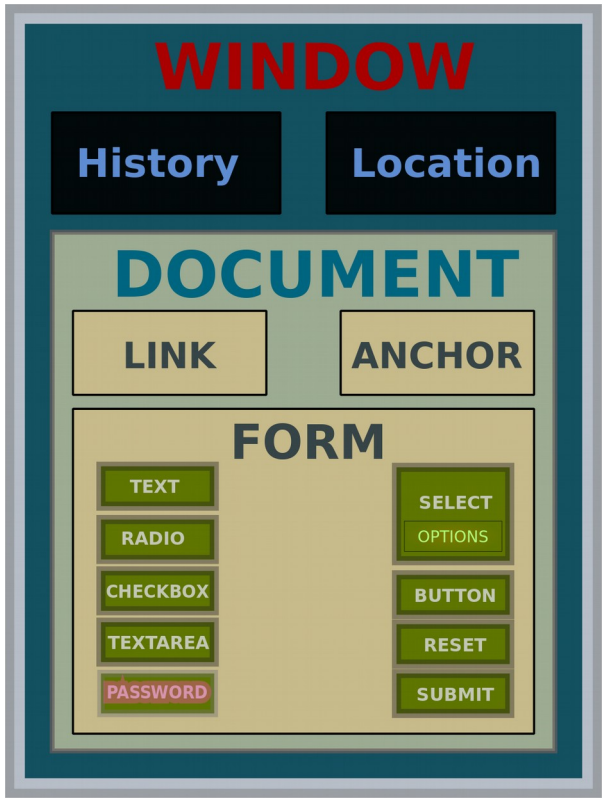
```
<html>
   <head>
      <title>Page of John
      Smith</title>
   </head>
   <body>
      <p>John Smith</p>
   </body>
</html>
```

XML:

```
<?xml version="1.0"
        encoding="UTF-8"?>
   <name>
      <first>John</first>
      <last>Smith</last>
   </name>
```

Infomation hierachies – Document Object Models (DOM)

- Comaprison between XML and HTML – different purpose:
  - HTML – specific purpose (formatting and visualization)
  - XML – no specific purpose – different information structures
- HTML Document Object Model - DOM
- XML Document Object Model – DOM



HTML Elements

- Tags, elements, attributes
- Document tree – types of nodes
- Element content – simple, mixed
- Document type – dictionary. HTML/XHTML/XML validation:
  - ***Document Type Definition (DTD), XML Schema***
- XML visualization in a web browser:
  - ***Cascading Style Sheets – CSS***
  - Interactivity – programming language ***JavaScript (EcmaScript)*** for excution of client side code in the browser

Basics of (X)HTML (1)

- XML markup and content – markup is enclosed between **<** and **>** (tags) or between **&** and **;** (entities), everyting other thsan that is **content**
- XML parser and user application – processes and analysis the markup and sends structured information to to user application
- Tags: ** **, ** **, ** **
- XML element – logical element in the XML document tree – simple or mixed content model:

** I amGeorge. **

Basics of (X)HTML (2)

- Attribute – key-value pair, included inside the tag:

```
<div id='15' style="color:red">
    Learn HTML for a day
</div>
```

HTML / XHTML декларации:

- **HTML5:** <!DOCTYPE html>
- **HTML 4.01:** <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
- **XHTML 1.0:** <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

Well Formatted HTML

- Root element of HTML Document should be **html.**
- HTML start with openeing and end with a closing tag.
- Tags should be properly nested (nested) – e.g.: ...Content...
- Attributes are inside the tag and always enclose their values in parenthesis.
- Tags and attributes are recommended to be in lowercase.
- Symbols **<**, **>**, **&**, **'**, **“** are changed to entities entities: **< > & ' "**

HTML 5 Base Template

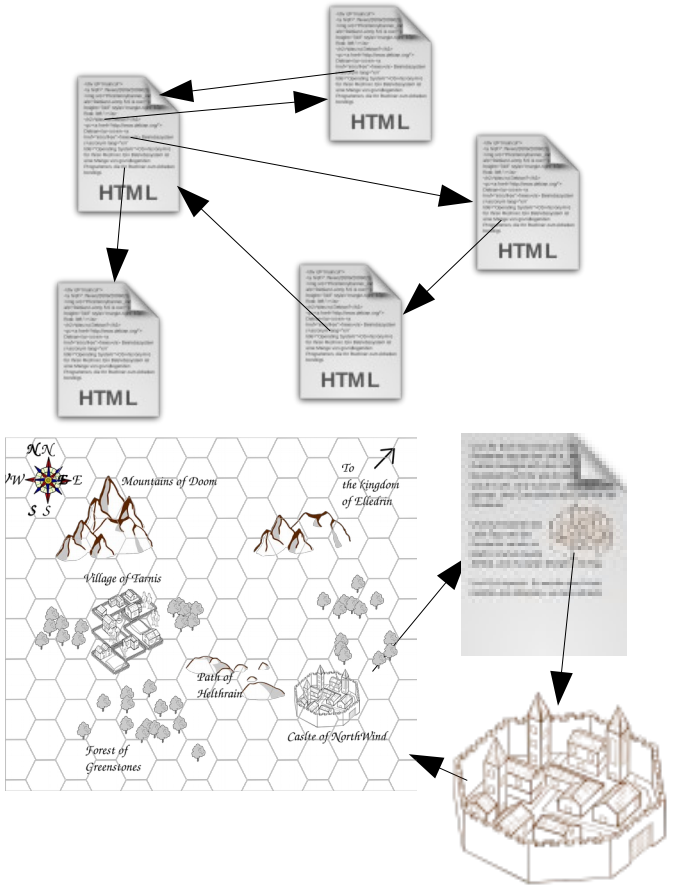
```
<!DOCTYPE html>
<html>
   <head>
      <title>Insert title here</title>
      <meta charset="UTF-8">
      <meta name="viewport"
          content="width=device-width, initial-scale=1.0">
      <style type="text/css">
         ... Example CSS ...
      </style>
   </head>
   <body>
       ... Example HTML ...
   </body>
</html>
```

HTML 5 Extended Template

```
<!DOCTYPE html>
<html>
   <head>
      <title>Insert title here</title>
      <meta charset="UTF-8">
      ...
   </head>
   <body>
      <!-- Enable HTML 5 elements in IE 7+8 -->
      <!--[if lt IE 9]>
         <script src="dist/html5shiv.js"></script>
      <![endif]-->
      ... Example HTML ...
   </body>
</html>
```

Hypertext & Hypermedia

- Hypertext is structured text that uses logical links (hyperlinks) between nodes containing text
- HTTP is the protocol to exchange or transfer hypertext
- Hypermedia - extension of the term hypertext, is a nonlinear medium of information which includes multimedia (text, graphics, audio, video, etc.) and hyperlinks of different media types (e.g. image or animation/video fragment can be linked to a detailed description.



Multipurpose Internet Mail Extensions (MIME)

- Different types of media are represented using different text/binary encoding formats – for example:
  - Text -> plain, html, xml ...
  - Image (Graphics) -> gif, png, jpeg, svg ...
  - Audio & Video -> mp3, ogg, webm ...
- Multipurpose Internet Mail Extensions (MIME) allows the client to recognize how to handle/present the particular multimedia asset/ node:

  Media Type Media SubType (format)

  Ex.: Content-Type: text/plain
- More examples for standard MIME types: https://developer.mozilla.org/en-US/docs/Web/HTTP/Basics_of_HTTP/MIME_types
- Vendor specific media (MIME) types: application/vnd.*+json/xml

Software Architecture – Definitions [1]

Almost everybody feels at peace with nature: listening to the ocean

waves against the shore, by a still lake, in a field of grass, on a windblown heath. One day, when we have learned the timeless way

again, we shall feel the same about our towns, and we shall feel as

much at peace in them, as we do today walking by the ocean, or stretched out in the long grass of a meadow.

— Christopher Alexander, The Timeless Way of Building (1979)

Software Architecture – Definitions [2]

According to **Dr. Roy Thomas Fielding** [Architectural Styles and the Design of Network-based Software Architectures, 2000]:

A **software architecture** is an abstraction of the **run-time** elements of a software system during some phase of its operation. A system may be composed of many levels of abstraction and many phases of operation,each with its own software architecture.

A software architecture is defined by a configuration of architectural elements - **components, connectors, and data** - constrained in their relaionships in order to achieve a desired set of architectural properties.

Software Architecture – Definitions [3]

According to **Dr. Roy Fielding** [Architectural Styles and the Design of Network-based Software Architectures, 2000]:

An **architectural style** is a coordinated set of architectural constraints that restricts the roles/features of architectural elements and the allowed relationships among those elements within any architecture that conforms to that style.

The primary distinction between **Network-based architectures** and software architectures in general is that communication between components is restricted to message passing, or the equivalent of message passing if a more efficient mechanism can be selected at runtime based on the location of components.

Architectural Properties

According to **Dr. Roy Fielding** [Architectural Styles and the Design of Network-based Software Architectures, 2000]:

- Performance
- Scalability
- Reliability
- Simplicity
- Extensibility
- Dynamic evolvability
- Cusomizability
- Configurability
- Visibility

- All of them should be present in a desired Web Architecture and REST architectural style tries to preserve them by consistently applying several **architectural constraints**

Service Oriented Architecture (SOA) – Definitions

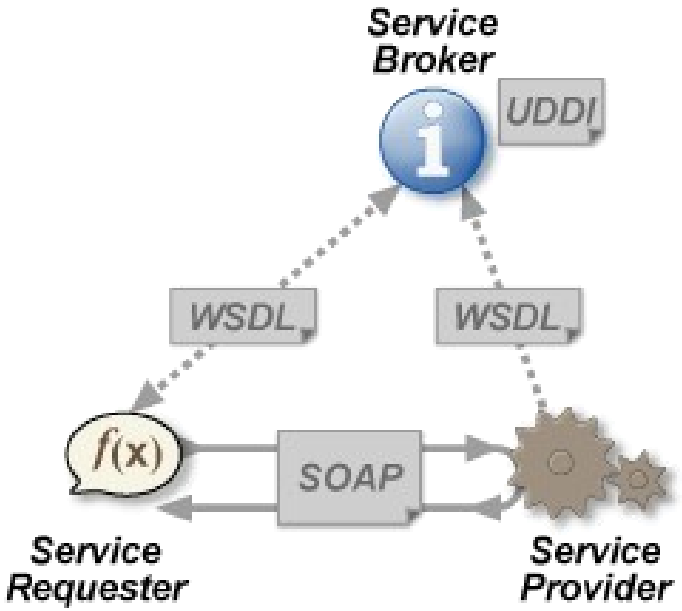
Thomas Erl: SOA represents an open, agile, extensible, federated, composable architecture comprised of autonomous, QoS-capable, vendor diverse, interoperable, discoverable, and potentially reusable services, implemented as Web services. SOA can establish an abstraction of business logic and technology, resulting in a loose coupling between these domains. SOA is an evolution of past platforms, preserving successful characteristics of traditional architectures, and bringing with it distinct principles that foster service-orientation in support of a service-oriented enterprise. SOA is ideally standardized throughout an enterprise, but achieving this state requires a planned transition and the support of a still evolving technology set

References: Erl, Thomas. Serviceorientation.org – About the Principles, 2005–06

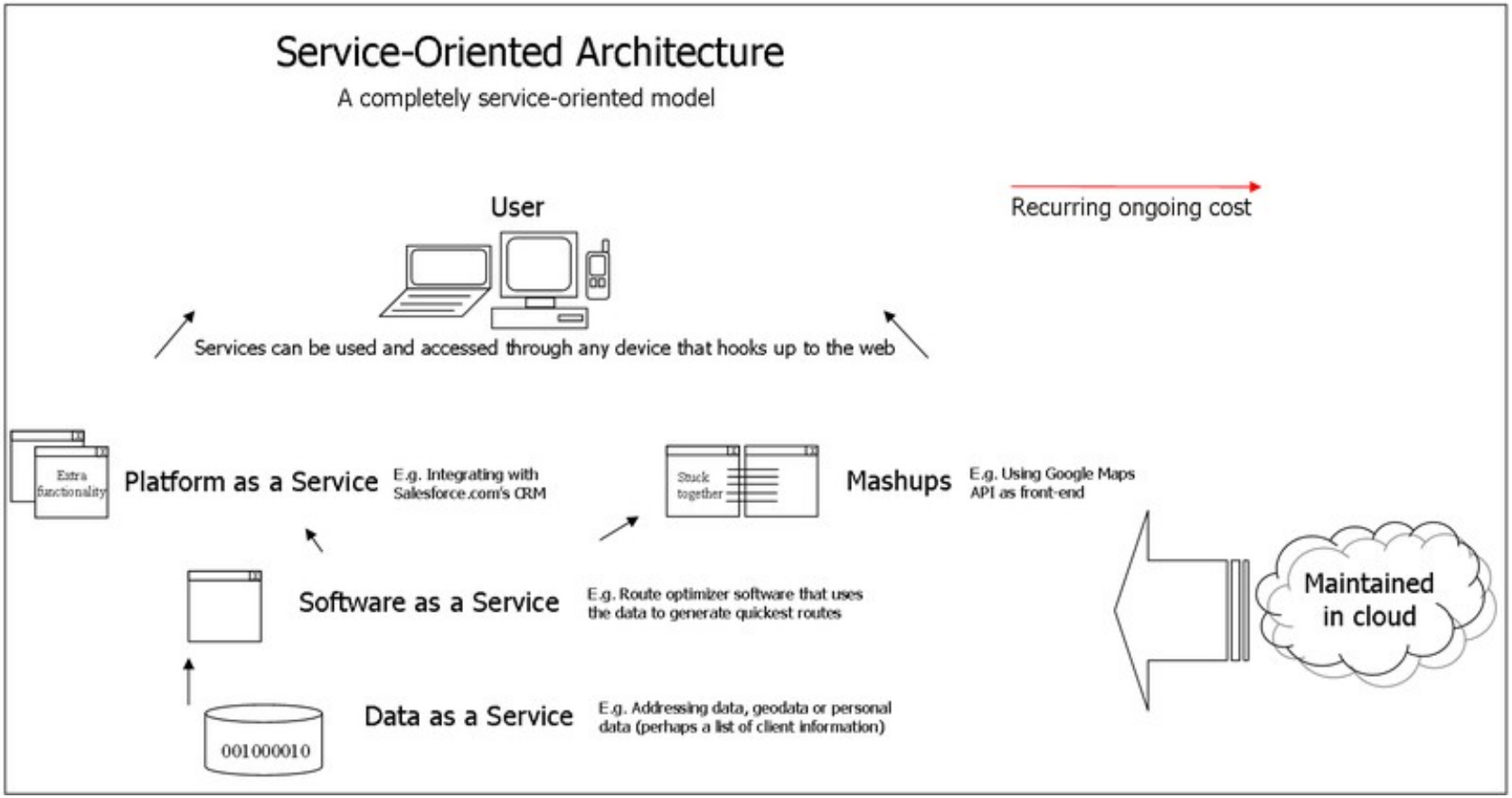
Classical Web Services - SOAP + WSDL

Web Services are:

- components for building distributed applications in SOA architectural style
- communicate using open protocols
- are self-descriptive and self-content
- can be searched and found using UDDI or ebXML registries (and more recent specifications – WSIL & **Semantic Web Services**)



Service Oriented Architecture (SOA). Mashups



Representational State Transfer (REST) [1]

- REpresentational State Transfer (REST) is an architecture for accessing distributed hypermedia web-services
- The resources are identified by URIs and are accessed and manipulated using an HHTP interface base methods (GET, POST, PUT, DELETE, OPTIONS, HEAD, PATCH)
- Information is exchanged using representations of these resources
- Lightweight alternative to SOAP+WSDL -> HTTP + Any representation format (e.g. JavaScript™ Object Notation – JSON)

Representational State Transfer (REST) [2]

- Identification of resources – URIs
- Representation of resources – e.g. HTML, XML, JSON, etc.
- Manipulation of resources through these representations
- Self-descriptive messages - Internet media type (MIME type) provides enough information to describe how to process the message. Responses also explicitly indicate their cacheability.
- Hypermedia as the engine of application state (aka **HATEOAS**)
- Application contracts are expressed as **media types** and [semantic] link realtions (**rel** attribute - RFC5988, "Web Linking")

[Source: http://en.wikipedia.org/wiki/Representational_state_transfer]

Hypermedia As The Engine Of Application State (HATEOAS) – New Link Header (RFC 5988) Example

```
Content-Length →1656
Content-Type →application/json
Link →<http://localhost:8080/polling/resources/polls/629>;
rel="prev"; type="application/json"; title="Previous poll",
<http://localhost:8080/polling/resources/polls/632>;
rel="next"; type="application/json"; title="Next poll",
<http://localhost:8080/polling/resources/polls>;
rel="collection"; type="application/json"; title="Polls
collection", <http://localhost:8080/polling/resources/polls>;
rel="collection up";  type="application/json"; title="Self
link", <http://localhost:8080/polling/resources/polls/630>;
rel="self"
```

Simple Example: URLs + HTTP Methods

Uniform Resource Locator (URL)	GET	PUT	POST	DELETE
Collection, such ashttp://api.example.com/comments/	**List the URIs and perhaps other details of the collection's members.**	**Replace the entire collection with another collection.**	**Create a new entry in the collection. The new entry's URI is assigned automatically and is usually returned by the operation.**	**Delete the entire collection.**
Element, such ashttp://api.example.com/comments/11	**Retrieve a representation of the addressed member of the collection, expressed in an appropriate Internet media type.**	**Replace the addressed member of the collection, or if it does not exist, create it.**	Not generally used. Treat the addressed member as a collection in its own right and create a new entry in it.	**Delete the addressed member of the collection.**

Source: *https://en.wikipedia.org/wiki/Representational_state_transfer*

Advantages of REST

- Scalability of component interactions – through layering the client server-communication and enabling load-balancing, shared caching, security policy enforcement;
- Generality of interfaces – allowing simplicity, reliability, security and improved visibility by intermediaries, easy configuration, robustness, and greater efficiency by fully utilizing the capabilities of HTTP protocol;
- Independent development and evolution of components, dynamic evolvability of services, without breaking existing clients.
- Fault tolerat, Recoverable, Secure, Loosely coupled

Richardson's Maturity Model of Web Services

According to **Leonard Richardson** [Talk at QCon, 2008 - http://www.crummy.com/writing/speaking/2008-QCon/act3.html]:

- Level 0 – POX: Single URI (XML-RPC, SOAP)
- Level 1 – Resources: Many URIs, Single Verb (URI Tunneling)
- Level 2 – HTTP Verbs: Many URIs, Many Verbs (CRUD – e.g Amazon S3)
- Level 3 – Hypermedia Links Control the Application State = HATEOAS (Hypertext As The Engine Of Application State) === **truely** RESTful Services

Web Application Description Language (WADL)

- XML-based file format providing machine-readable description of HTTP-based web application resources – typically RESTful web services
- WADL is a W3C Member Submission
  - Multiple resources
  - Inter-connections between resources
  - HTTP methods that can be applied accessing each resource
  - Expected inputs, outputs and their data-type formats
  - XML Schema data-type formats for representing the RESTful resources
- But WADL resource description is **static**, while resources change dynamically – REST anti-pattern

RESTful Patterns and Best Practices

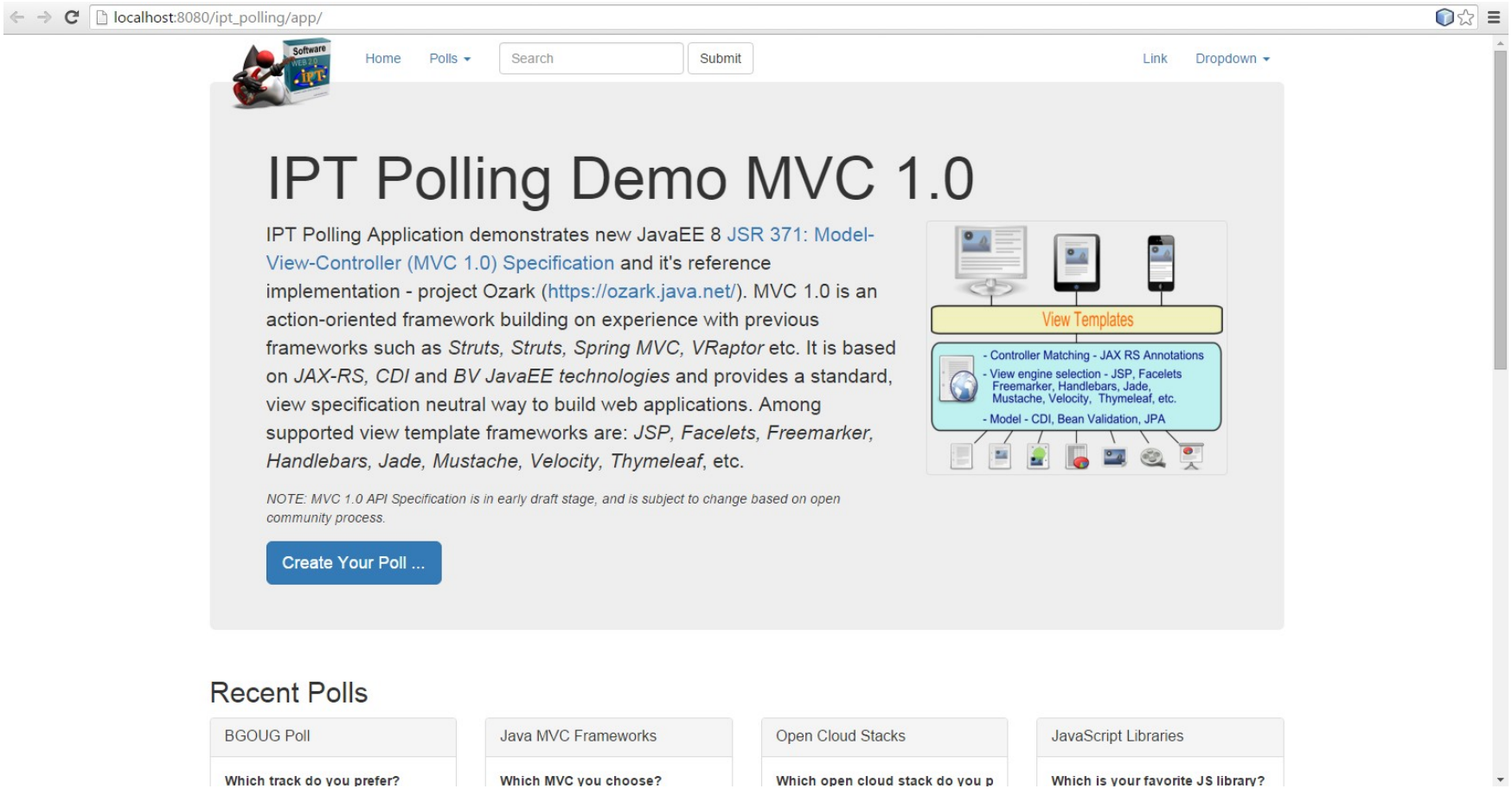
According to **Cesare Pautasso** [http://www.jopera.org/files/SOA2009-REST-Patterns.pdf]:

- Uniform Contract
- Content Negotiation
- Entity Endpoint
- Endpoint Redirection
- Distributed Response Caching
- Entity Linking
- Idempotent Capability

REST Antipatterns and Worst Practices

According to **Jacob Kaplan-Moss** [http://jacobian.org/writing/rest-worst-practices/]:

- Conflating models and resources
- Hardcoded authentication
- Resource-specific output formats
- Hardcoded output formats
- Weak HTTP method support (e.g. tunell everything through GET/POST)
- Improper use of links
- Couple the REST API to the application

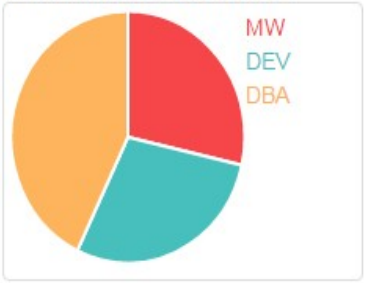

localhost:8080/ipt_polling/app/
Home
Polls
Search
Submit
Link
Dropdown
IPT Polling Demo MVC 1.0
IPT Polling Application demonstrates new JavaEE 8 JSR 371: Model-View-Controller (MVC 1.0) Specification and it's reference implementation - project Ozark (https://ozark.java.net/). MVC 1.0 is an action-oriented framework building on experience with previous frameworks such as Struts, Struts, Spring MVC, VRaptor etc. It is based on JAX-RS, CDI and BV JavaEE technologies and provides a standard, view specification neutral way to build web applications. Among supported view template frameworks are: JSP, Facelets, Freemarker, Handlebars, Jade, Mustache, Velocity, Thymeleaf, etc.
NOTE: MVC 1.0 API Specification is in early draft stage, and is subject to change based on open community process.
Create Your Poll ...
View Templates
- Controller Matching - JAX RS Annotations
- View engine selection - JSP, Facelets Freemarker, Handlebars, Jade, Mustache, Velocity, Thymeleaf, etc.
- Model - CDI, Bean Validation, JPA
Recent Polls
BGOUG Poll
Which track do you prefer?
Java MVC Frameworks
Which MVC you choose?
Open Cloud Stacks
Which open cloud stack do you p
JavaScript Libraries
Which is your favorite JS library?

localhost:8080/ipt_polling/app/

Recent Polls

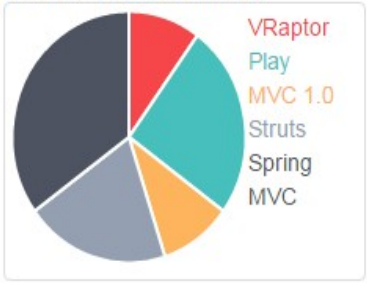
BGOUG Poll

Which track do you prefer?



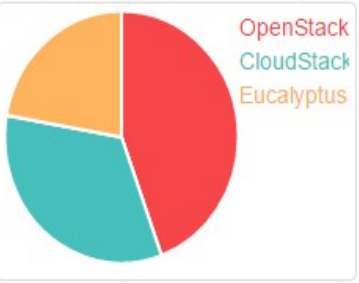
Java MVC Frameworks

Which MVC you choose?



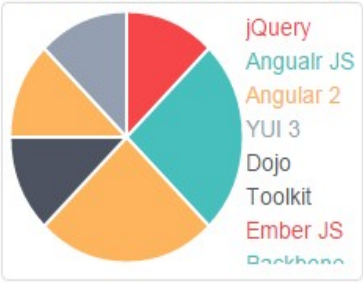
Open Cloud Stacks

Which open cloud stack do you p



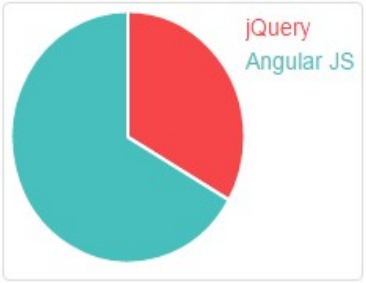
JavaScript Libraries

Which is your favorite JS library?



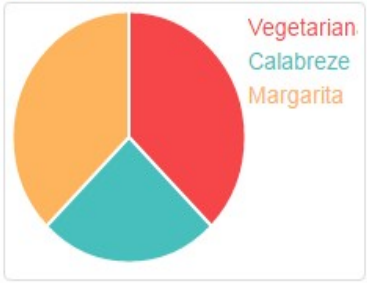
JavaScript Libraries

Which is your favorite JS library?

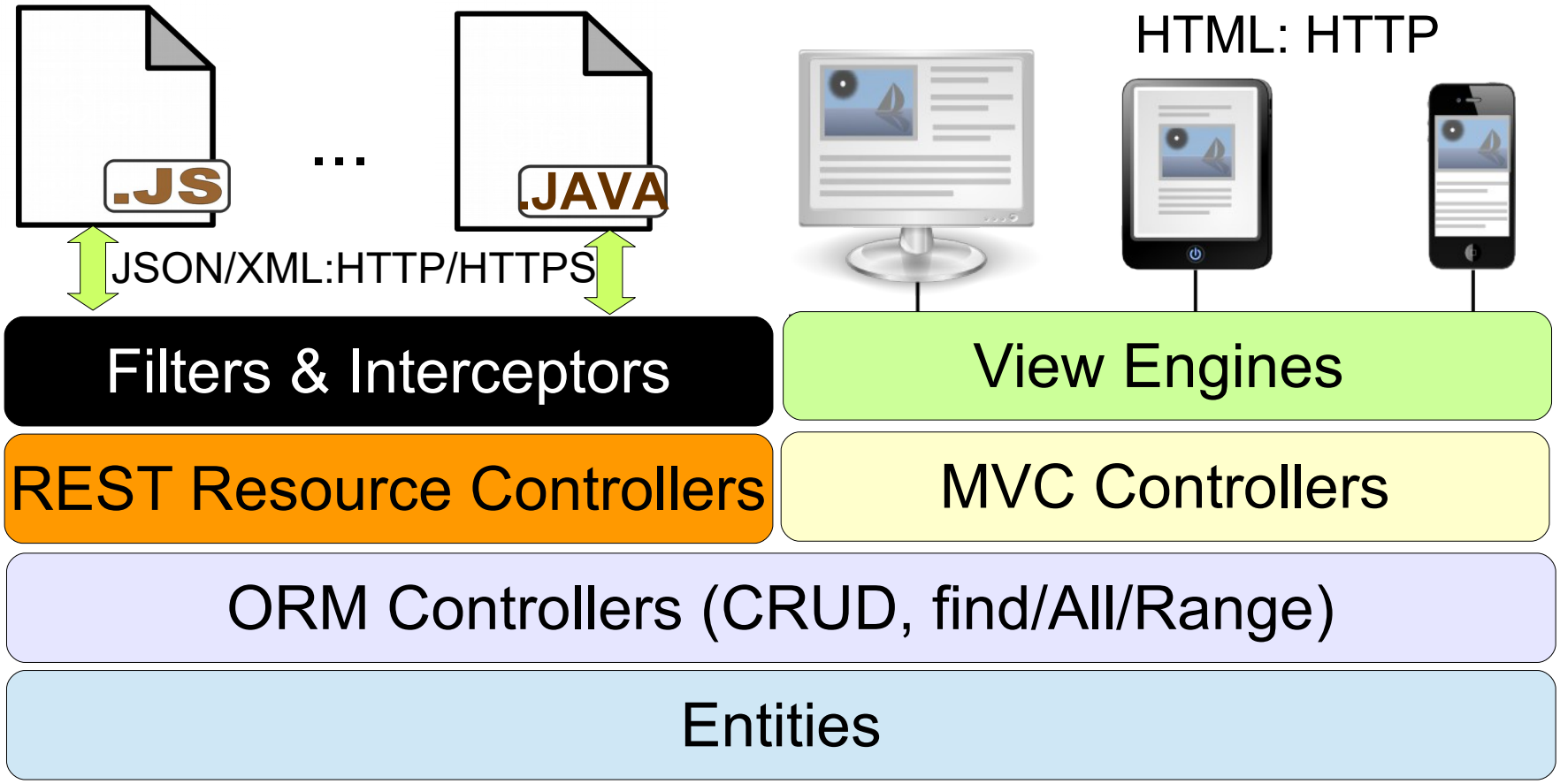


Pizzas

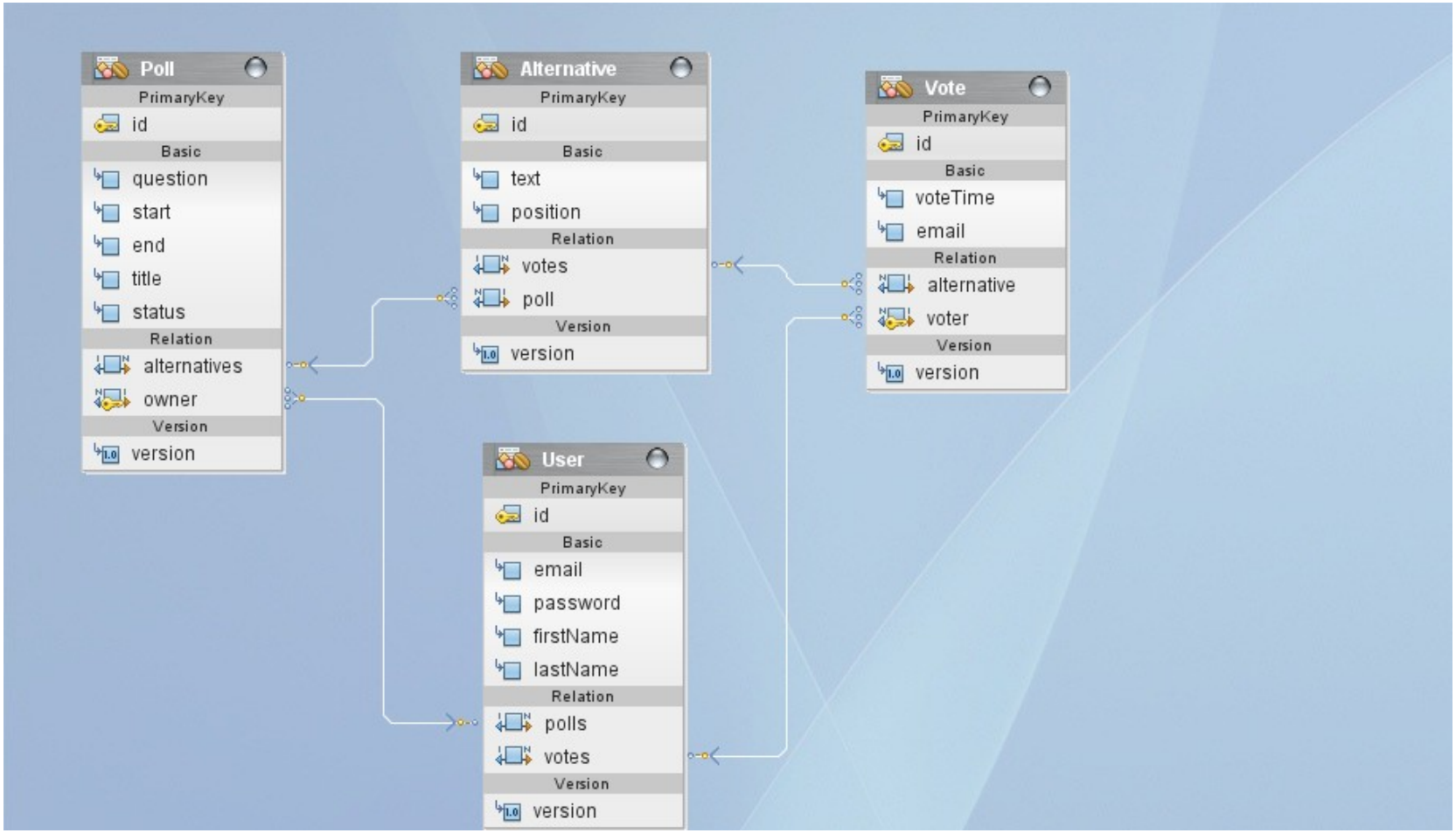
What is your favorite pizza?



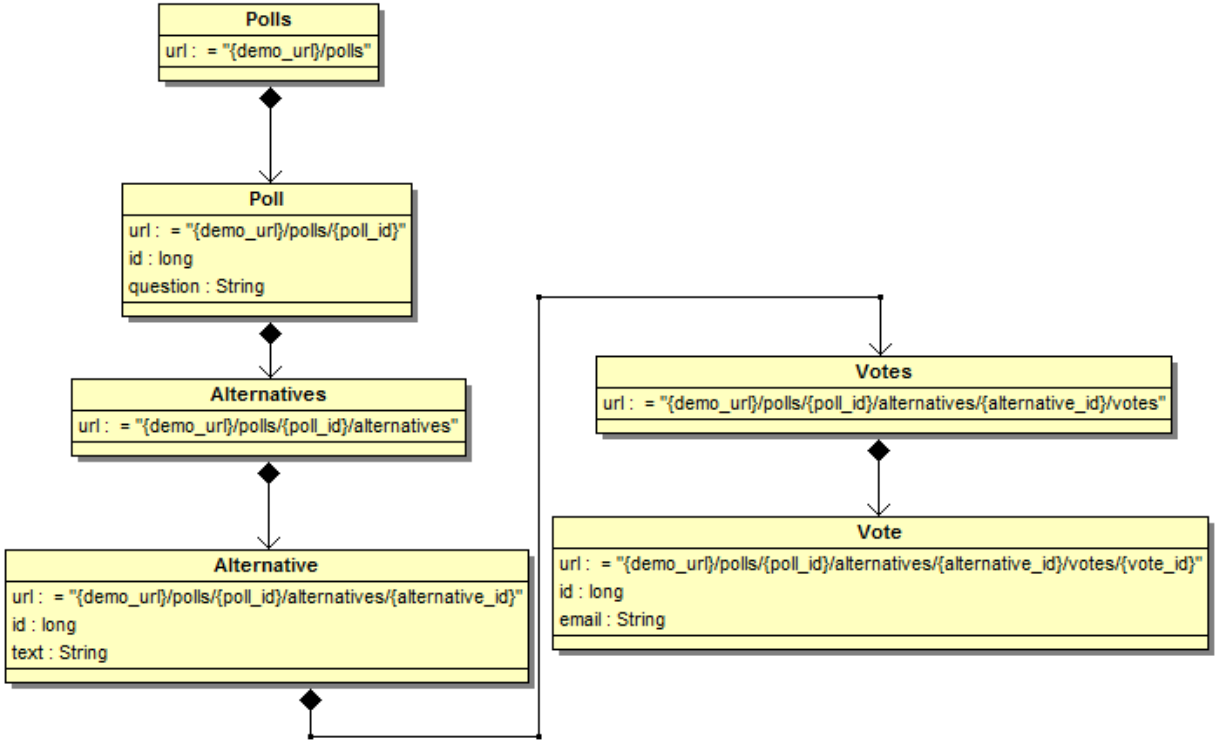
N-Tier Architectures



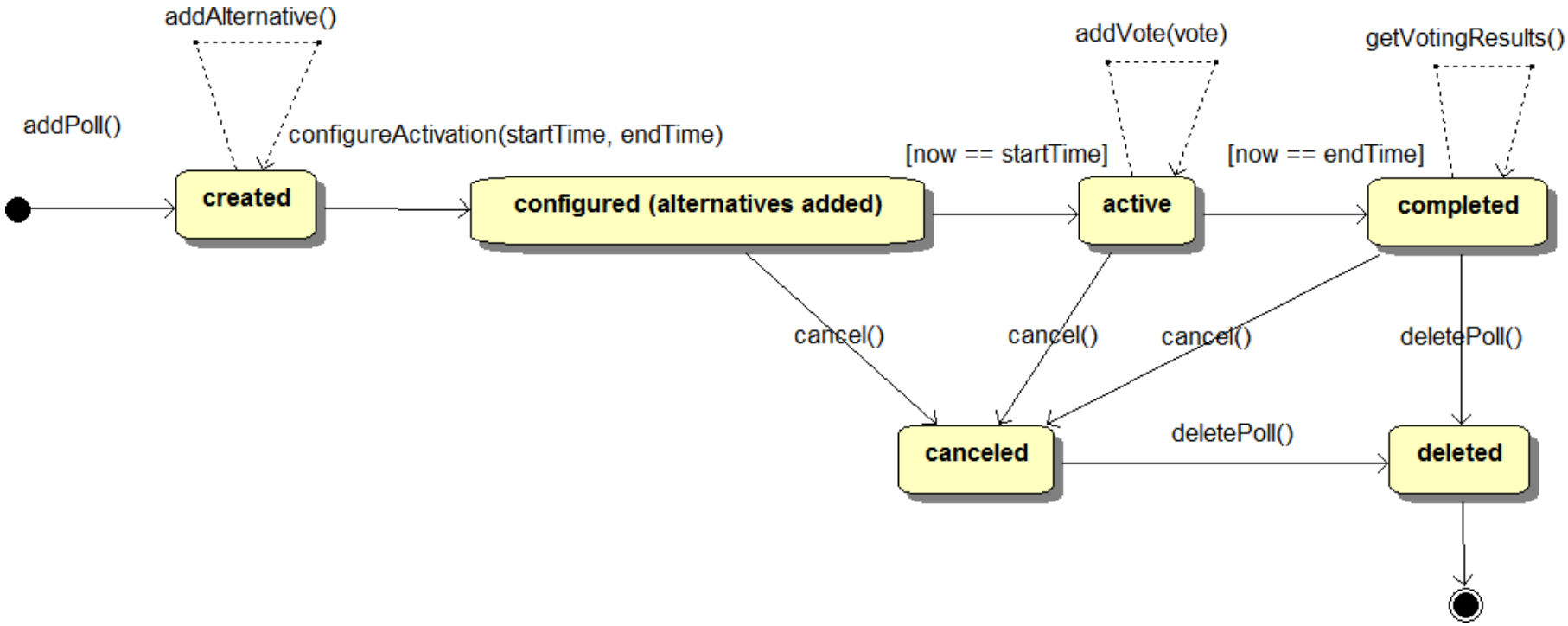
IPT Polling Demo MVC 1.0 Data Architecture



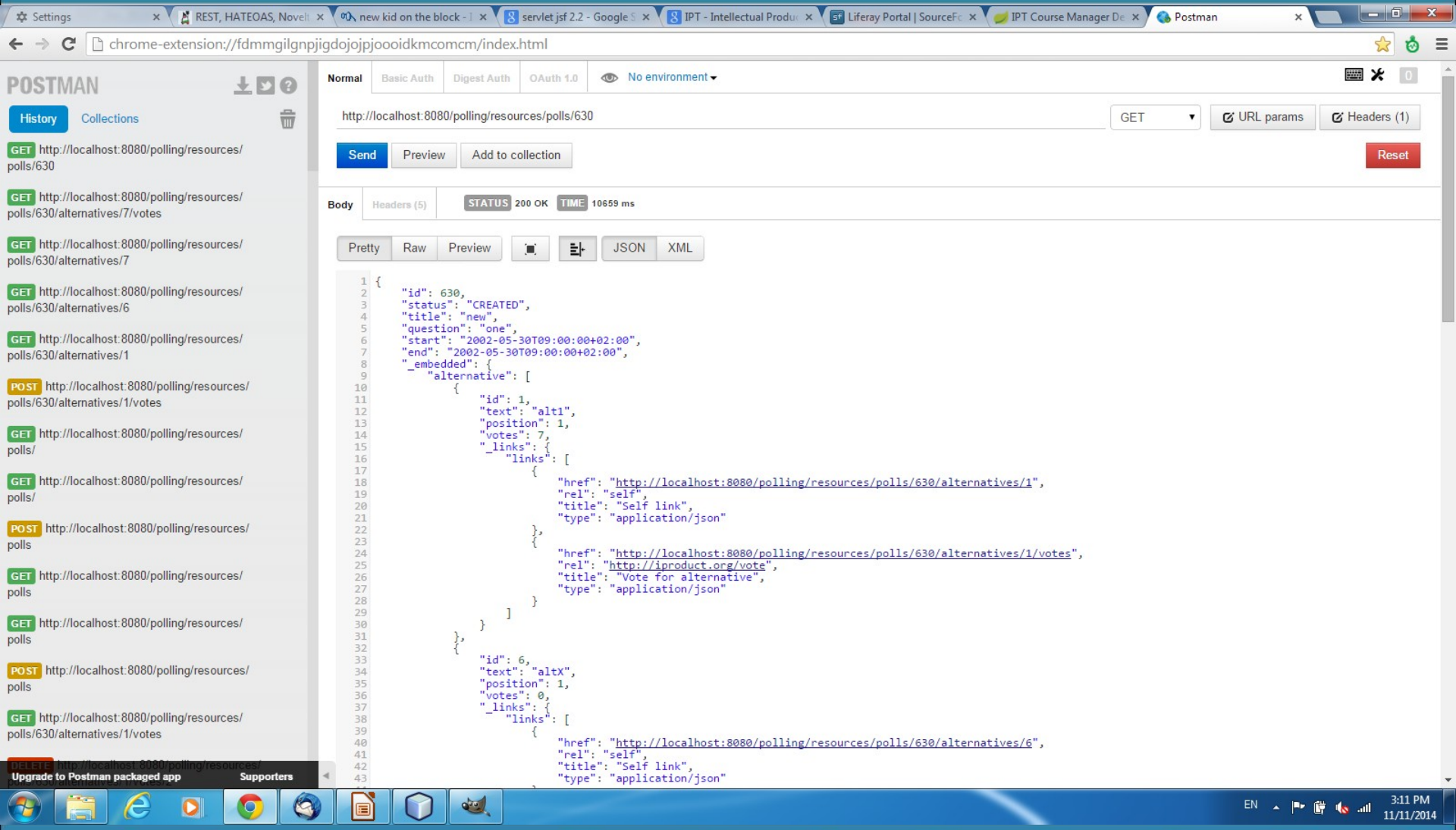
IPT Polling Demo Resources



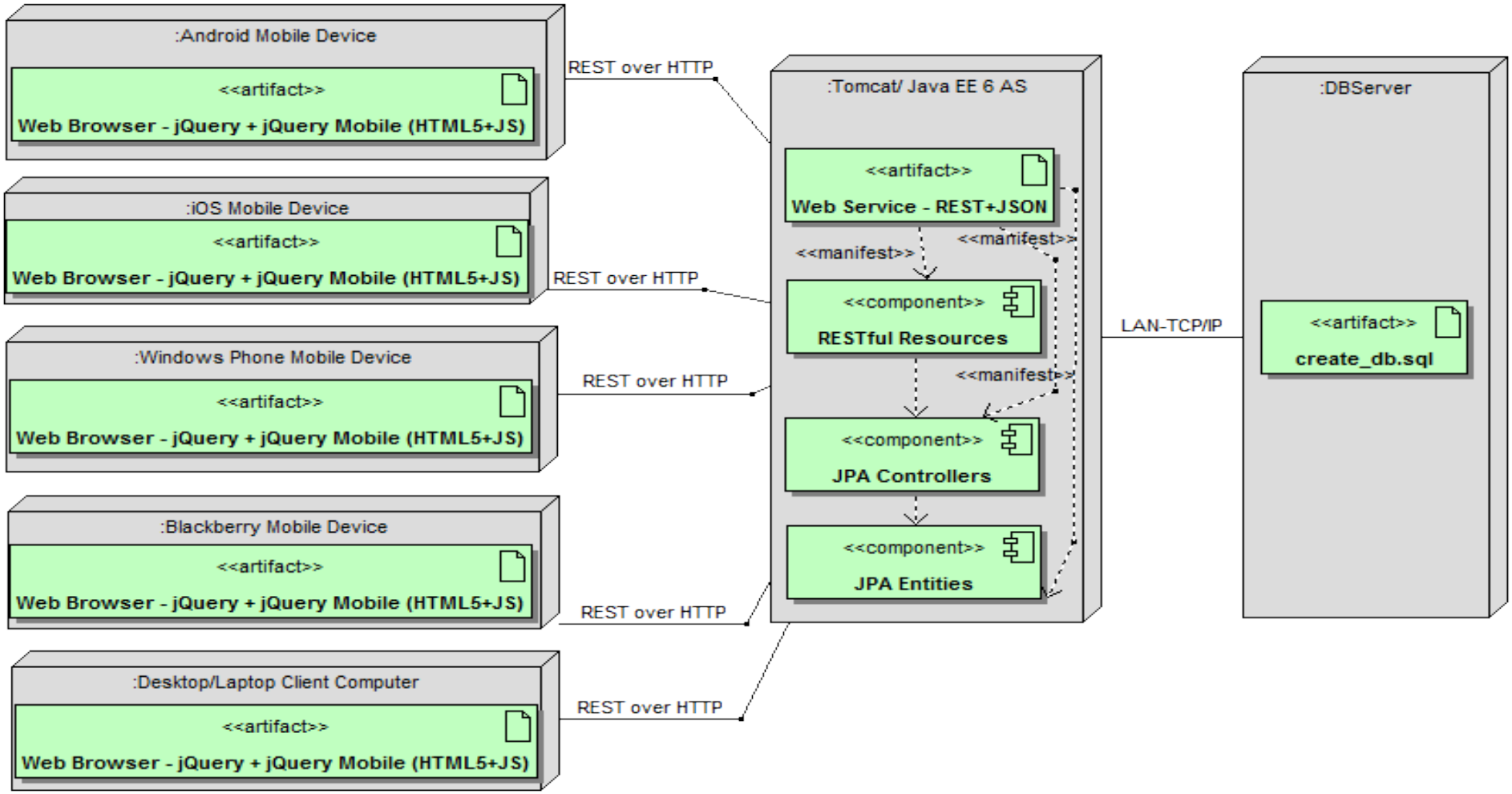
IPT Polling Demo – Poll Resource States



IPT Polling Demo



Proposed Architecture: jQuery Client + RESTful Backend



References

- R. Fielding, Architectural Styles and the Design of Networkbased Software Architectures, PhD Thesis, University of California, Irvine, 2000
- Fielding's blog discussing REST – http://roy.gbiv.com/untangled/2008/rest-apis-must-be-hypertext-driven
- Representational state transfer (REST) in Wikipedia – http://en.wikipedia.org/wiki/Representational_state_transfer
- Hypermedia as the Engine of Application State (HATEOAS) in Wikipedia – http://en.wikipedia.org/wiki/HATEOAS
- JavaScript Object Notation (JSON) – http://www.json.org/
- JAX-RS 2.0 (JSR 339) Reference Implementation for building RESTful Web services – http://jersey.java.net/

Thank's for Your Attention!

Trayan Iliev

CEO of IPT – Intellectual Products & Technologies

http://iproduct.org/

http://robolearn.org/

https://github.com/iproduct

https://twitter.com/trayaniliev

https://www.facebook.com/IPT.EACAD

https://plus.google.com/+IproductOrg